行政院國家科學委員會專題研究計畫 成果報告

數位音樂典藏之資料探勘與智慧型檢索技術
研究成果報告(完整版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 94-2422-H-004-003-
執 行 期 間 ：94 年 03 月 01 日至 95 年 05 月 31 日
執 行 單 位 ：國立政治大學資訊科學系

計 畫 主 持 人：沈錳坤

計畫參與人員：碩士班研究生-兼任助理：邱士銓、陳文婷、魏綾音、江孟芬、彭冠誌

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，2 年後可公開查詢

中 華 民 國 95 年 12 月 21 日

行政院國家科學委員會補助專題研究計畫 ☒ 成 果 報 告
□期中進度報告

數位音樂典藏之資料探勘與智慧型檢索技術

計畫類別：☑ 個別型計畫　　□ 整合型計畫
計畫編號： NSC　94 － 2422 － H － 004 － 003 －
執行期間：94 年 3 月 1 日 至 95 年 5 月 31 日

計畫主持人： 沈 錳 坤
共同主持人：
計畫參與人員： 邱士銓、陳文婷、江孟芬、魏綾音、彭冠誌

成果報告類型(依經費核定清單規定繳交)：□精簡報告　☒完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
☒涉及專利或其他智慧財產權，□一年☒二年後可公開查詢

執行單位：國立政治大學資訊科學系

中　華　民　國　95　年　5　月　31　日

數位音樂典藏之資料探勘與智慧型檢索技術

摘要

本計畫的研究重點即為利用音樂探勘技術，研究數位音樂典藏中的智慧型擷取技術。數位音樂典藏的擷取方式包括以後設資料(metadata)查詢、音樂內容擷取, 音樂曲風查詢, 相關回饋, 音樂瀏覽與個人化音樂推薦等，以有助於使用者方便地擷取典藏的數位音樂。在本計畫中，我們主要目的將研究利用音樂探勘中的音樂樣式探勘將音樂分段，以發展音樂瀏覽(Music Browsing)技術。

音樂分段在研究音樂分析相關的領域是很重要的研究題目。音樂的分段可以提供作音樂結構分析、音樂瀏覽、音樂內容查詢與音樂摘要等應用。本計劃的研究目的就是對音樂作自動分段，以幫助使用者能快速瀏覽音樂的內容。因此，我們針對音樂的主題作主題式的分段。

音樂的主題是取決於作曲者的動機，動機是構成音樂主題的基本因素。為了能夠以主題為基礎作音樂分段，我們必須找出決定音樂主題的因素。動機會有規則性的出現在整首音樂當中，所以我們可以利用動機出現的規則來探勘音樂的動機。

我們提出一個以主題對音樂作分段的方法，總共分為四個主要的步驟。第一，我們從原始的音樂資料擷取出主旋律的部分。第二，將主旋律做粗略分段。我們利用探勘 Non-trivial 重複樣式的技術[17]，來找出粗略段落。第三，從粗略段落中探勘動機。我們利用 Stein 所提出來的動機變化規則，修改傳統探勘重複序列的方法，做動機的探勘。最後，我們利用探勘出來的動機對主旋律作精細分段。我們針對 MIDI 音樂檔案利用提出來的方法，實做出一個系統，找出音樂的主題段落。

先前研究在評估實驗結果時，多採用 Precision 與 Recall 去評估實驗的結果。然而，這樣的評估方法並不能表現出實驗結果與正確答案之間的相似程度。所以我們提出新的評估方法，根據實驗結果與正確答案之間的相似程度來評估實驗的準確率。根據實驗結果顯示，我們的方法準確率約 65%。

Data Mining and Intelligent Retrieval Techniques for Digital Music Archives

Abstract

In this project, we investigated the data mining techniques for intelligent retrieval of digital music archive. Music browsing is one of the retrieval mechanisms for music archive. Music segmentation is one of the important design issues for music browsing. Music segmentation can be utilized for music structure analysis, music browsing, content-based music retrieval, and music summarization. In this project, we proposed a music segmentation method based on the music theme to provide users the capability to browse music segments by theme.

Motives, the concepts of the composer, are the basic elements of music themes. Music themes were constructed by motives. In order to segment music by themes, we have to discover motives. Most motives repeated in the music by some motivic treatment rules. Therefore, motives can be discovered by these rules.

We proposed the theme segmentation method. There are four steps. Firstly, we extract main melody from original music. In the second step, rough segments are generated from main melody by mining non-trivial repeating patterns. Then, motives are detected from rough segments. We modify the mining algorithm for discovering frequent patterns by applying motivic treatment rules proposed by Stein. Finally, we segment main melody based on the generated motives. Moreover, a system for segmentation of music in MIDI format was implemented.

Concerning the effectiveness evaluation of music segmentation, precision and recall are used in previous research. We proposed an effectiveness measure and corresponding algorithm to evaluate the accuracy of music segmentation. Experimental results show that our proposed music segmentation method achieves 65% accuracy.

目錄

圖目錄

表目錄

第一章

概論

1.1 動機

「數位典藏國家型科技計畫」的目的在於妥善保存國家的文化資產，推廣精緻藝術的流傳與品賞，強健文化的傳承與發展，並鼓勵資訊與知識的分享。在目前典藏的文物中，常見的媒體形式多以文字、影像與視訊為主體，以音樂為主題的典藏資料庫並不多。

然而台灣在音樂方面，從國樂、南北管、原住民音樂、閩南民謠、客家民謠乃至校園民歌都是豐富的文化資產。國樂中，由董榕森先生作曲的「陽明春曉」是耳熟能詳的梆笛演奏曲。而 1996 年亞特蘭大奧運主題曲「反璞歸真」即源自於馬蘭地區的阿美族人郭英男的「老人飲酒歌」。日據時期由鄧雨賢先生作曲、李臨秋先生作詞的「望春風」則為家喻戶曉的閩南民謠。921 地震之後，旅美的大提琴家郭虔哲即在美國卡內基音樂廳演奏閩南民謠「望春風」、「補破網」、「雨夜花」來賑災募款。60 年代陳達的恆春民謠「思想起」，一把月琴、一生艱辛更是台灣經濟發展過程中的重要文化資產。在客家民謠方面，無論是老山歌、山歌仔、平版調，都是質樸勤奮的客家人之寫照。而 70 年代由於時代背景而風行一時的校園民歌，隨著時光的流逝，也逐漸進入我們的回憶中。這些音樂都伴隨著台灣的發展，以音樂的方式訴說著我們文化發展的歷程。

與其他形式的媒體類似，數位音樂典藏的技術研究議題包括智慧財產權與隱私權管理機制、數位典藏資料庫技術與多媒體處理技術。隨著 MP3 及 P2P 的盛行，智慧財產權管理機制在數位音樂中扮演重要的角色。與文字影像視迅等媒體不同，人類對音樂的認知是透過聽覺的方式而非視覺的方式。因此人類在表達音樂擷取的條件時，相對地比較困難。音樂擷取技術不僅有助於一般的使用者擷取數位音樂典藏，也有助於建置、維護與管理典藏資料的專業使用者。例如，數位典藏加值服務的專業配樂師有搜尋音樂的需求，以方便為以 60 年代台灣鄉村風格為場景的廣告做配樂。

本計畫的研究重點即為利用音樂探勘技術，研究數位音樂典藏中的智慧型擷取技術。數位音樂典藏的擷取方式，最直接的方式是以後設資料(metadata)查詢擷取。與音樂有關的後設資料包括曲名、作曲者、樂派、歌詞或以文字描述的曲風等。但如同其他媒體形式的後設資料，後設資料必須透過專業人員的加註(Annotation)，也因此會有解讀(Interpretation)的問題。因此，音樂內容擷取(Content-based Retrieval), 音樂曲風查詢(Music Style Retrieval), 相關回饋(Relevance Feedback), 音樂瀏覽(Music Browsing)與個人化音樂推薦(Personalized Music Recommendation)等功能將有助於使用者方便地擷取典藏的數位音樂。

其中，音樂瀏覽提供音樂的分類索引，讓使用者以瀏覽的方式擷取音樂。此外，每首音樂也可以摘要(Summarization)的方式呈現，以方便使用者不必循序的播放而直接地聆聽音樂。這將有助於瀏覽諸如交響樂等長度長的音樂。在本計畫中，我們主要將研究利用音樂探勘技術，發展音樂瀏覽技術。

一般作家在寫作時，會將文章依據主題利用段落的方式分開。讀者在瀏覽文章時，可以依據章節或主題來閱讀。相較於文字資料，多媒體資料中的影片或音樂，如果在製作時沒有明顯的段落，使用者可能必須將整個資料從頭看過或聽過，才能瀏覽到所需要的片段。舉例來說，貝多芬第九號交響曲「合唱」，最有名的演奏版本長度為 74 分鐘，愛樂人尋找自己喜歡的片段時，可能需要花費不少時間才能找到。因此，多媒體資料的分段，有助於使用者以分段的方式瀏覽多媒體資料。

分段的好處不止於提供使用者瀏覽的需求，我們進一步可以利用分段的結果產生出多媒體內容的摘要(Summary 或 Thambnailing)。此外，分段的技術也可以建立查詢所需的索引，用在以內容為基礎的多媒體查詢(Content-based multimedia retrieval)上。

一般多媒體的資料量都非常龐大，如果以人工對多媒體資料做分段，需要花費很多的人力與時間。因此，在多媒體的研究領域，已經有許多視訊分段的相關研究[23]。相較於視訊，音樂分段的研究在最近才興起[22]。最近幾年開始有研究學者[2][3][22][40]針對音樂資料做分段，有一些研究做音樂分段的目的是找出比單一一個音符階層高的音樂單位，藉此可以做高階的分析，如：合聲等。這些研究純粹找出許多分段點(Boundaries)。所以，兩個連續的音符中間，如果特徵有很大的落差，可能就是一個分段點。這些研究並沒有針對音樂的內容做分段。如果要達到主題式瀏覽的目的，可能要對音樂的內容做分析，如：

音樂主題的分析等。

音樂資料是具有結構性的，作曲家在作曲時會有動機，最後再由動機發展出主題，將許多個主題串起來組成完整的音樂。因此，我們可以根據音樂的結構做主題的分段，以提供使用者做主題的瀏覽。我們希望可以發展一套方法，對音樂內容做主題式的分段，讓使用者可以對音樂的主題做瀏覽。

1.2 背景

音樂的特徵包含旋律(Melody)、節奏(Rhythm)、音色(Timbre)、力度(Dynamic)、和聲(Harmonic)、織體(Texture)和曲式(Form)等元素，以特定的方式組合時，而產生獨特的效果與特色。音樂家為了將這些訊息記錄下來，讓其他人也能夠演奏出來，開始有人發展出記譜的方法。透過樂譜，演奏的人可以將上述音樂物理狀態，盡可能重現出來。接下來我們簡單介紹基本音樂記譜方法所記錄的資訊。

音樂記譜的基本單位為音符，一個音符代表一個樂器所發出的一個特定聲音。幾個音符構成一個小節，許多小節構成完整的音樂。每一個音符最基本的特徵為音高(Pitch)與音長(Duration)。數個音高決定音樂的旋律，數個音長決定音樂的節奏。

對於音高來說，Musical Scale 定義音高的連續關係，最常見的以八度音階(Octave)為主，也就是八個音符間隔構成七個音階，這個連續的關係會循環不變。以自然音階(Diatonic)為例，圖 1.1 以五線譜(staff)表示八個音的連續關係。音樂中，有一種定音高的準則稱為十二平均律，這是目前使用最廣泛的音高準則。這個規則會將一個八度音階切分成十二個半音。以鋼琴鍵為例，如圖 1.2 所示，箭號所指的琴鍵為兩個相差八度音階的“do”。以每一個琴鍵為半音，橫跨在八度音階中的黑白琴鍵共有十二個。因此，有一種稱為 Naturally Occurring Half Step 的現象發生。Naturally Occurring Half Steps 會發生在自然音階的第三與第四個音之間，以及第六個與第七個音之間。通常在音與音之間，會存在一個半音階(Semitone 或 half steps)，也就是兩個鋼琴白鍵中間所夾的黑鍵，兩個音的間距為兩個半音。依據十二平均律，半音也是目前一般音樂中，最小的音程單位。然而，在 Naturally occurring half steps 的地方，兩個音的距離只有半個音程。以 MIDI 記錄音高的表示法來看，圖 1.1 的“do re mi fa sol la ti do”，會表示為 <60, 62, 64, 65, 67, 69, 71, 72>。在<64, 64>與<71, 72>的地方為半音階，其他皆為全音階。兩個音符之前的音高差，我們稱為 Pitch interval。以

自然音階的 do 跟 mi 為例，兩者的 Pitch interval 為 2 個全音。

do re mi fa sol la ti do

1 1 1/2 1 1 1 1/2

圖 1.1：自然音階的音程關係。

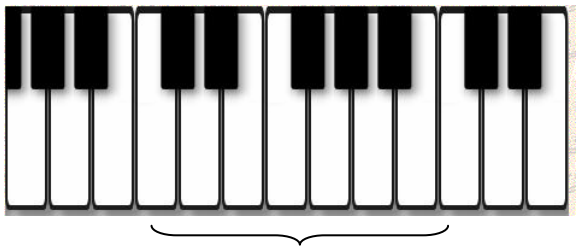


圖 1.2：一個八度音階的鋼琴鍵分布。

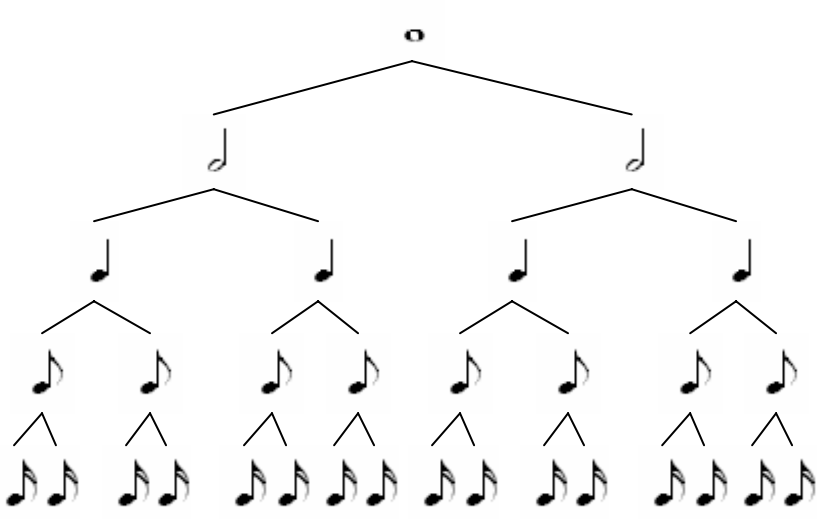


圖 1.3：音長倍數關係表示圖。

接下來，我們介紹音長的關係。每個音長與單位時間，擁有固定的倍數關係；此外，音樂演奏時，也會分別針對不同音樂的需求，訂出不同的單位時間。圖 1.3 為樂理中，表示音長關係的樹狀圖。每一個相鄰的階層，父節點都為子節點音長的 2 倍。在樹狀圖中的根節點，我們稱為全音符，代表音長為四拍，由根節點分成兩個二分音符，代表的音長為

兩拍。接下來的四分音符、八分音符與十六分音符的音長也都保持相同的關係。休止符的時間長短一樣有相同的關係。還有一種“附點”的音符，表示為原來音長的 1.5 倍。以兩拍的二分音符為例，如果二分音符加上附點，所代表的音長為 2+1 拍。

我們進一步看音樂中音符包含的資訊，每一個音符有一個開始發聲的時間與結束發聲的時間，音長為兩者的時間差。此外，兩個連續音符的開始發聲時間間距，稱為 Inter-Onset Interval(IOI)；兩個連續音，第一個音的結束發聲時間到第二個音開始發聲的時間間距，稱為 Onset-Offset Interval(OOI)，也就是所謂的休止符。

根據 Stein[41]，作曲家在譜寫一首音樂時，以動機(Motive)為基本單位。在上述音樂特徵中，動機所含的必要元素為旋律以及節奏，動機至少包含一組特有的音程(Pitch Interval)與一組特有的音長，音程可以構成旋律，音長構成節奏。音程是連續的音，兩兩相鄰音的音高差；節奏是連續的音，以不同音長先後排列的關係。貝多芬第五號交響曲“命運”第一樂章的主要動機，命運之神的敲門聲，就是最初四個音“Sol Sol Sol Mi~”，不斷在整個交響樂中的第一樂章出現。為了不讓音樂只是單調的重複同一個動機，作曲家會安排動機以不同的改變形式重複出現在音樂中。以交響曲“命運”第一樂章的動機來看，有時整體動機音比較高，有時音比較低，但是動機的結構都與最初四個音一樣。動機的變化會在 4.1 詳細的描述。

在作曲的程序中，完整的音樂作品是以動機為依據，連續不斷的變化構成音樂的主題；多個相關的主題可以組合產生音樂的主旋律。因此，我們可以將動機探勘出來後，以音樂的動機為線索切割音樂段落，並且將每一個段落視為同一首音樂中的不同主題。Lartillot[25][26]指出，先前研究顯示，人類是因為記住音樂的“前後結構”關係，因此可以在動機改變時辨認出來。音樂旋律也常依據某些易於讓人類記憶的規則而製作出來。

1.3 MIDI 格式

一般來說，數位音樂儲存的方式有兩種，第一種為高階的 Symbolic 的儲存方法，另外一種為低階的 Acoustic 的儲存方法。通常，Acoustic 所包含的音樂資訊是最為完整的，包含所有人類可以感知的資訊，而 Symbolic 的儲存方式，只能紀錄完整的旋律與節奏以及粗略的音色。然而，分析動機只需要旋律與節奏兩項最主要的特徵，要從 Acoustic 儲存方法分析出旋律與節奏是很困難的；而 Symbolic 儲存方式可以直接取得這兩項音樂特徵。最常

見的 Symbolic 儲存方法為 MIDI 檔案格式，同時也是最容易取得資料的檔案格式。因此，本計劃所處理的音樂資料是以 MIDI 檔案格式儲存的。

表 1.1：Head Chunk 格式。

Head Chunk				
Chunk Type	Data Length	Data		
4 bytes (ASCII code)	4 bytes (32 bits binary)	6 bytes		
		16 bits	16 bits	16 bits
MThd	<length>	<format>	<tracks>	<division>

表 1.2：Track Chunk 格式。

Track Chunk		
Event Type	Data Length	Data
4 bytes (ASCII code)	4 bytes (32 bits binary)	<length> bytes (binary data)
MTrk	<length>	<delta time><event>,<delta time><event>, …

表 1.3：一個 MIDI Event Data 的記錄方式。

Delta time$_1$	Event$_1$
Delta time$_2$	Event$_2$
Delta time$_3$	Event$_3$
…..	
Delta time$_{k-1}$	Event$_{k-1}$
Delta time$_k$	Event$_k$

MIDI 的全名是 Musical Instrument Digital Interface，常見的 MIDI 有三種不同的格式。我們一般最常見的格式為 GM(General MIDI)，另外兩種為 GS(General Standard)與 XG(Extended General MIDI)。我們接下來以 GM 的 MIDI 檔案格式為主做介紹，檔案架構由一個＜Header chunk＞與多個＜Track chunk＞所組成。如表 1.1 所示，在 Header chunk 裡包括了播放格式(format)，音軌的數量(tracks)，演奏的基本時間單位(division)等訊息。如表 1.2 所示，在一個 Track chunk 內，紀錄許多 MIDI 事件(Event)，每個 Track chunk 代表一個 MIDI track。每一個事件前，會記錄一個事件發生的時間(Delta-time)，如表 1.3 所示。

通常，一個音符包含兩個事件的發生，第一，Onset 事件，代表音符開始發音；第二，Offset 事件，代表音符結束發音。在 Onset 與 Offset 事件中，都會記錄音符的音高與音量。根據標準的 MIDI 格式，音高的範圍是 0 到 127，我們常使用的 C 大調音階，也就是自然音階如圖 1.1 所示，在 MIDI 所記錄的音高資訊為<60, 62, 64, 65, 67, 69, 71>，每一個單位都代表半音(Semitone)。此外，音量的範圍也是 0 到 127。Onset 與 Offset 兩個事件的時間間距，代表該音符的音長。所以，由這兩個事件所記錄的資訊，我們可以知道一個音符開始發音的時間、結束發音的時間、音高、音量與音長。由於 MIDI 資料是以 Binary 格式儲存，我們必須先將 MIDI 透過 Parse 處理後，在分析 MIDI Event 的資料。

MIDI 的製作，通常是透過製作者以演奏的方式輸入到電腦內記錄下來。音高以半音為單位，音長以 Tick time 為單位。但是，樂譜上兩個同樣拍子的音符，實際在演奏中不會完全相同。所以，我們在探勘音樂的資料之前，必須以 MIDI 檔案所記錄的拍子時間單位，做音長的校正(Quantization)。

1.4 報告架構

接著我們介紹本篇報告的架構。在第二章，我們將介紹 Music segmentation、Thematic extraction、Summarization 與 Mining frequent subsequence 等相關研究。第三章，我們介紹我們計畫主要的方法，包含 MIDI 資料擷取、主旋律擷取方法、兩種粗略分段方法、兩種動機探勘方法，以及最後以主題分段的方法。第四章，介紹我們實作所需要的技術與方法。第五章，介紹我們的實驗設計、實驗的評估方法，以及實驗結果。第六章，根據我們實驗的方法做討論，以及未來研究可能的方向。

第二章

相關研究

2.1 音樂分段

Spevak[40]等人指出，音樂的分段是將獨立的音符以時間結構的關係結合起來。目前研究主要的作法是從音樂的主旋律找出許多組的分段點(Boundary) ，每一個段落都是由前後兩個分段點所決定，因此將主旋律分成許多局部的片段(Local boundary segmentations)。有時候，音樂中有些部份是找不到完整的分段點，所以分段的結果可能不會涵蓋整段音樂。

音樂的小片段可以幫我們分析出比較高階的音樂特徵，如：對比、和聲等。通常音樂可`以利用音長、音高資訊來決定分段點，不考慮較高階的音樂資訊[3][15]。一些研究學者[2]針對民俗音樂(或民謠 Folksong)做分解時，考慮歌詞的押韻，因為歌曲中的歌詞與旋律會相互影響。

另一方面來看，音樂可以依照主題分成許多的段落。比如說：古典音樂的奏鳴曲式可以分為主題、主題發展、主題再現、終止四大段落。這種與音樂實質內容有關的段落形式，並不能單純以分段點來決定。目前很少研究是針對音樂的段落來處理，音樂學已經有許多相關的資訊與音樂的分段有關係，如：和弦進程 (Chord progression)可以讓我們分辨出一個完整的段落，我們一般常聽見的民歌，在利用吉他彈唱時，都會固定的和弦進行。這樣的方法並不能反映出音樂主題的內容，只是單純的將音樂切割開。

Generative Theory of Tonal Music

Lerdahl 與 Jackendoff[27]在 1983 年提出 “Generative Theory of Tonal Music” (GTTM)，Lerdahl 等人將調性音樂(Tonal music)於人類直覺上的意義，具體的描述。GTTM 主要用於從音樂的表面形式(Surface)，找出隱含於內的結構。GTTM 主要能分析的音樂是以主音音樂(Homophony)為主，主音音樂是由一個主旋律與外加伴奏所組成。GTTM 是目前較具體

的音樂分析理論，具有相當的參考價值。首先，這個理論將音樂以群組(Group)的方式來分析。群組的方式指的是把主音音樂分解成一群一群的小單位，並且具有階層的關係。圖 2.1 為一個 GTTM 所群組分析後的結果，所有的群組都必須符合被上層群組階層包含的關係。

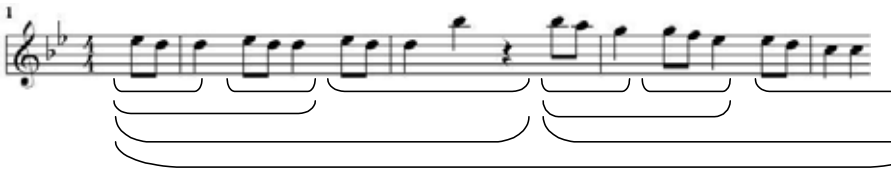


圖 2.1：GTTM 群組範例。

GTTM 利用主音音樂在演奏或演唱時，需要換氣的部分來作為分段的依據，而這些分段點，大都是音長較長、休止符出現或是音高的起伏改變時的部分。除此之外，GTTM 還利用相鄰音之間的音長或休止當作距離(Proximity)，以及利用各個音相似度(Similarity)，來決定群組的範圍。如圖 2.2 所示，以相同形狀來表示相似音，圖形之間的距離代表音長或休止。因為相鄰關係與相似關係不同的考量，會改變群組的方式。舉例來說，圖 2.2 的 B 中間的部分有一個圓形與方形距離相近，我們這時候可以選擇不同的比重考量相鄰與相似的關係。

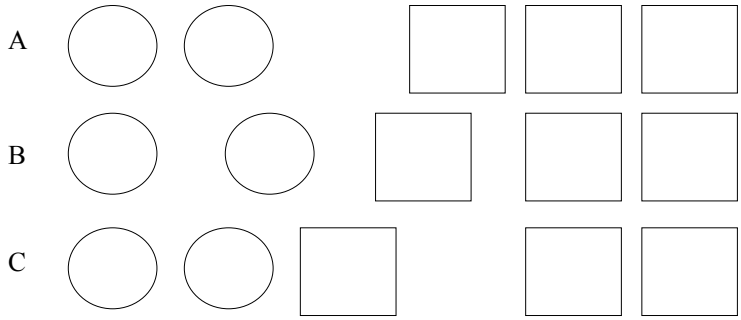


圖 2.2：相鄰關係與相似關係的群組。

Grouper

Temperley 等人[42]利用音樂的 Duration 資訊(Inter-Onset Interval、Offset-Onset Interval)與 Metric 資訊(Beat List)，找出可能的分段點。這個方法從長度為 n 個音符的音樂中，分別去分析 2^n 個可能的分段方法，然後求出分段最好的結果。Temperley 利用 GTTM 的規則，來評估分段好壞。大致來說，這些規則會讓分段點在 IOI 與 OOI 間距很大，而且分段點剛

好在 Beat 上的結果得到較好的評估排序。

Local Boundary Detection Model(LBDM)

Cambouropoulos[3][4]利用 GTTM 所描述的相鄰與相似的關係，提出 Local Boundary Detection Model(LBDM)。LBDM 所能處理的音樂與 GTTM 一樣，是針對主音音樂的主旋律。由主旋律分別計算出音程差(Pitch interval)、音長(Interonset interval, IOI)及休止(rest-計算目前的 Onset-time 到上一個 Offset-time 的間距)三個特徵序列。LBDM 計算三個序列得出“Boundary strength value”，數值序列。數值序列中的高峰(Peak)部分，就是群組或分段的邊界。LBDM 訂出兩個計算 Boundary strength value 的規則，“Change Rule (CR)”與“Proximity Rule(PR)”。CR 針對一個特徵序列，計算出 “Degree of change”。假設我們取用三種音樂的特徵值 k，k∈{pitch, ioi, rest}，每一個特徵值序列以 $\text{Pro}_k = [x_1, x_2, ..., x_n]$表示，所有 Pro_k 所包含的元素都大於等於零。在 x_i 與 x_{i+1} 的 Degree of change r 計算公式如下：

$$r_{i,i+1} = \frac{|x_i - x_{i+1}|}{x_i + x_{i+1}} \text{，iff } x_i + x_{i+1} \neq 0$$

$$r_{i,i+1} = 0 \text{，iff } x_i = x_{i+1} = 0$$

PR 則是找出特徵序列中，可能的分界點，分界點由 Strength value 高的決定。對於一個特徵 k，可以計算出 $V_k=[v_1, v_2, ..., v_n]$，其中 $v_i = x_i \cdot (r_{i-1,i} + r_{i,i+1})$。加權整合每個特徵所分別計算出來的 V_k，可以決定最後可能的邊界。

Data-Oriented Parsing

Bod[2]針對民俗音樂分段提出 Data-Oriented Parsing(DOP)的方法，利用一個已知分段的音樂資料庫來做 learning，建立一個機率分布。然後在利用 learning 的結果對其他音樂做分段。Bod 利用 Natural Language 的方法套用到音樂的資料上，主要利用三種 Probabilistic grammar technique 做 learning，分別是 Treebank Grammar Technique，Markov Grammar Technique 與 Extended Markov Grammar Technique with DOB Technique。此外，這個方法考慮民俗音樂歌詞押韻的方式，同樣也是以 Natural Language 的技術處理。

Perceptual Analysis

Jian 等人[22]提出以人類聽覺認知對音樂訊號做分段，有別於傳統聲音訊號的分段。他們利用 Roughness、Periodicity pitch 與 Loudness 三項特徵，找出分段點。Roughness 包含有音樂的音色與節奏的資訊；Periodicity pitch 包含音高的資訊；Loudness 包含音量的資訊。與前面所提到的 LBDM 所用的原理相同，特徵值中，如果有劇烈變化的地方，就可能是分段點，例如：在音色變換的時候，Roughness 值會有劇烈的變化。不過，這個方法並不適用於對主題做分段。在交響曲中，主題常常被不同樂器輪流演奏出來。舉例來說，貝多芬第六號交響曲“田園”第一樂章的第一個主題，是由小提琴、豎笛與雙簧管三種樂器輪流演奏出來。如果以音色變化來做分段，會將第一主題分成許多小段。

Repeating Pattern based on LBDM

Chen[8]提出一個應用 LBDM 方法的研究，對整首音樂先分解出一個個片段。然後，以片段為單位，找出重複出現的片段。Chen 進一步修改 LBDM 的方法，使分解片段的效果，更貼近人類的感知。

Structural Analysis

Chai[7]針對 Acoustic 音樂資料，找出音樂的重複結構。音樂結構的分析分為五個主要的步驟，步驟如下。第一，Feature extraction，以 Frame 為單位切割音樂訊號，並且擷取每一個 Frame 的特徵值。對每一個 Frame 使用 Autocorrelation 的方法做特徵擷取，擷取出來的音樂特徵為該 Frame 所代表的音高。第二，Pattern matching，以固定的連續 Frame 個數組合成許多有重疊(overlap)的片段，利用 Local Alignment 的方法，找出各別重疊的片段在音樂序列中重複出現的情形。第三，Repetition detection，由前一個步驟的結果探勘出重複序列。第四，Segment merging，將性質相近的連續數個片段合成一個。最後，結構標示(Structure labeling)，將重複出現的段落做標示。因此，一首音樂在做完結構分析後，如圖 2.3 所示，我們可以得知圖中的音樂結構為 ABAB 的形式。

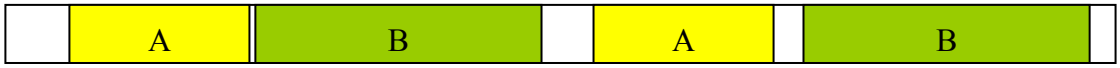


圖 2.3：一首音樂的重複結構分析結果，A 與 B 是重複出現的部份。

2.2 主題萃取

Meek[32][33]提出 Thematic Extraction 的問題，從音樂的資料中，找出能夠代表音樂的 Keywords 或動機。這些 Keywords 是從 Intervallic 的旋律中，找出重複出現的樣式。Meek 提出一個 Melodic Motive Extractor(MME)演算法，可以自動從音樂中找出重複出現的樣式。由於 Meek 所採用的樣式比對是以 Exact matching 的方法，重複樣式會包含許多沒有音樂意義的樣式。因此，Meek 訂出一些可以過濾樣式的方法，用以減少樣式的數量。

MME 輸入的音樂資料是以音符的形式，每一個音符包含開始發音的時間(Onset Time)，結束發音的時間(Offset Time)與音高(Pitch)，以及最長樣式長度限制 W。原始音樂的資料以長度為 *W* 的 Sliding Window 擷取資料，然後比對擷取出來的資料是否有重複出現的樣式。樣式主要是依靠音高的資訊做比對，開始與結束發音的時間用來做進一步的過慮以及最後所有 Patterns 的排序。最後輸出的結果是長度 2~*W* 的樣式，所有的樣式會比較樣式的 Duration，Rhythmic consistency，Position 與 Register 等特徵值來做排序。

2.3 音樂摘要

Music summarization 目的是從單一的音樂資料中，擷取出具有代表性的段落，藉此可以提供瀏覽音樂內容的功能。一些相關研究分別針對 Acoustic 音樂資料與 Symbolic 音樂資料，提出 Music summarization 的作法。

Hirata[15]提出針對 Symbolic 音樂資料，透過使用者互動的 Music summarization 系統。這個系統主要是幫助使用者能快速的製作出一首音樂的 Summarization，透過電腦先篩選出可能需要的片段，再讓使用者去做選擇。他們利用 GTTM 的方法，先將音樂分段，然後將所有分段表示成一個 Time span tree。接著，利用代表各個音樂分段的 Time span tree 分枝，作為計算相似度的依據，對所有音樂分段做分群。然後透過使用者互動的方式，去掉不符合需求的群組。最後，利用剩下的群組產生出 Music summarization。

Logan[29]等人提出從 Acoustic 音樂資料產生 Summary 的方法。第一，從訊號資料擷取出 Mel-cepstral 特徵序列，Mel-cepstral 是由語音辨識研究領域發展出來的特徵表示方法。接下來，以固定長度切割特徵序列成許多 Frames。第二，對所有的 Frames 做 Buttom-up clustering 或透過 Traning Hidden Markov model 的方法，將相似的 Frames 設定成同一個標籤(Label)。最後，從 Frames 中，挑出最常出現的標籤標示的 Frame 為 Key phrase。Xu 等人[49]改進 Logan 等人的方法，除了 Mel-cepstral 的特徵值以外，還多考慮 Spectral power 與 Amplitude envelope 兩種特徵值。

Bartsch[1]擷取出音樂訊號的 Chroma-based 特徵值，並且利用 Similarity matrix 紀錄特徵值比對的結果，以用來產生 Summary 的結果。方法總共分為五個步驟。第一，Frame segmentation，Bartsch 利用 Beat tracking 的技術，先找出音樂的節拍，然後利用接拍的資訊將音樂訊號分成許多 Frames。第二，Feature calculation，計算每個 Frame 是屬於哪一個音高種類(Pitch class)。Chroma-based 的特徵值將音高區分為 12 個種類，每個 Frame 透過計算後，分到 12 個種類裡。第三，Correlation calculation，比較任兩個 Frame 所屬的種類是否相同，並且以一個 Similarity matrix 記錄比較結果。第四，Correlation filtering，從 Similarity matrix 的資料，計算出 Time-lag matrix。最後，從 Time-lag matrix 中，找出重複出現的音樂片段。

Chai[6]針對 Acoustic 音樂資料中，做音樂的結構分析。利用音樂結構分析的結果，在對音樂做 Music summarization。接著，Chai 提出三種用來從結構分析結果選擇 Music summarization 的策略。第一，Section-beginning，發生在音樂一開始的片段為可能的 Music summarization。第二，Section-transition，發生在兩種重複片段中間的片段為可能的 Music summarization。最後，Multiple-phrase，選擇重複片段中，發生在音樂最前面的兩個片段，各取一半結合起來作為 Music summarization。

2.4 音樂樣式探勘

Musical Pattern Discovery (MPD)是最近興起的資訊科學研究領域，主要是對音樂做動機分析(Motivic analyses)。音樂的動機時常會做改變，如：不同音高的動機，但是人類還是可以知覺的出來。以前動機分析的研究大都將旋律與節奏分開處理，甚至省略節奏的資訊。不過，依據音樂學家分析音樂動機在音樂中變化的規則，旋律與節奏是必須同時考慮的。

圖 2.4 為巴哈所做的創意曲前三小節，矩形標出音樂動機出現的位置。第一聲部的 A、

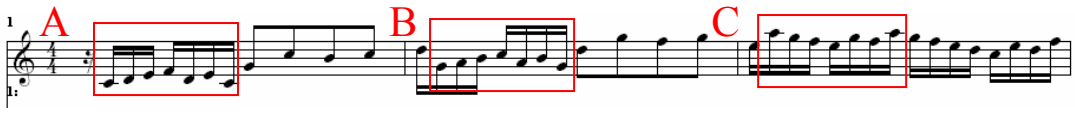


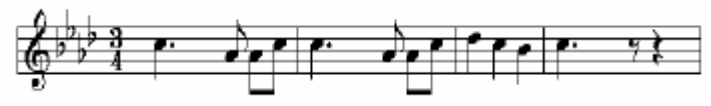
圖 2.4：巴哈 C 大調二部創意曲 BWV 001。

B、C 分別是動機原形、動機音程重複(Pitch Interval Repeat)、動機反向進行(Contrary motion)。純粹比較音高，A、B、C 三個皆不相同，如果比較音程(Interval)，A 與 B 音程相同，C 與 A、B 不相同。然而我們可以很明顯的看到，動機的節奏在 A、B、C 三個部分都是保持不變的。如果在探勘動機的過程，我們只考慮旋律或節奏，可能會找不到動機。音樂動機有重複出現的特性，並且含有時間先後的關係，相似於資料庫裡的**重複序列樣式**(Repeating sequence pattern)。

2.4.1 探勘精確重複樣式

Hsu[16][17]提出兩種探勘精確重複樣式(Mining Exact Repeating Pattern)演算法，找出單聲部音樂中的重複樣式。第一種演算法是以 Dynamic programming based 的方法，找出一首音樂中精確重複樣式。主要有三個步驟，需要給定 Minimum support。第一，將音樂旋律建立出一個 Correlative matrix，記錄旋律中重複的位置，以利於找出重複樣式。如圖 2.5 所示，將音符重複出現的地方記錄下來。圖 2.5 中，Correlative matrix 第一列代表第一個音 C6 在其他位置出現的情形；第二列代表第二個音 Ab5 在其他位置出現的情形。第二列中，Ab5 第二次出現的位置，由於前面一個音 C6 也重複出現，因此記錄 2，以代表目前有一組音連續出現的情形，2 代表這一組音目前的長度。其他列的計算依此類推，空白的部分代表沒

有任何符合的音。第二，從 Correlative matrix 的資訊產生所有可能的候選重複樣式。第三，計算所有留下候選重複樣式的出現次數，保留出現次數超過 Minimum support 的重複樣式。



	C6	Ab5	Ab5	C6	C6	Ab5	Ab5	C6	Db5	C6	Bb5	C6
C6	—			1	1			1		1		1
Ab5		—	1			2	1					
Ab5			—			1	3					
C6				—	1			4		1		1
C6					—			1		1		1
Ab5						—	1					
Ab5							—					
C6								—		1		1
Db5									—			
C6										—		1
Bb5											—	
C6												—

圖 2.5：建立 Correlative Matrix 的範例[17]。

Hsu[16]提出的第二種方法，是以 Apriori based 演算法，同樣找出精確重複樣式。主要有兩個步驟。第一，從音樂旋律中找出所有長度為 1 的重複樣式，並且記錄個別重複樣式的出現次數與位置。以旋律“C D E F C D E C D E F”為例，我們可以記錄長度為 1 的重複樣式：{“C”,3,(1,5,8)}, {“D”,3,(2,6,9)}, {“E”,3,(3,7,10)}, {“F”,2,(4,11)}。第二，利用前一個步驟產生出來長度為 n 的重複樣式，利用 Join 的方式，組合產生出長度為 n+1 的重複樣式。以前一個步驟長度為 1 的樣式，如：{“C”,3,(1,5,8)}, {“D”,3,(2,6,9)}，我們可以 Join 出{“C D”,3,(1,5,8)}；{“E”,3,(3,7,10)}與{“F”,2,(4,11)}可以 Join 出{“E F”,2,(3,10)}。接著不斷重複第二個步驟，直到無法產生新的重複樣式。

Meredith 等人[34]利用空間向量的觀點，從多聲部的音樂(Polyphonic music)中，找出精確重複樣式。Meredith 將音長忽略不計，純粹依據音高與音符開始發聲時間為空間的座標，計算出所有點與點之間的向量。將有相同向量群組在一起，一個向量的群組群組內的點，都可以依據此向量，同時找到對應的點，也就是重複出現的樣式。Meredith 所提出的方法有四個主要步驟。第一，將音樂資料，依照音高與發音時間為座標，轉換到座標空間上。第二，建立出一個矩陣計算旋律中任一點到其他點的向量。音樂旋律以空間的關係表達，每一個點為一個單一的音符。依據空間的關係，我們可以建立一個矩陣，藉此計算出每一個點與其他任一點的向量關係。第三，將計算出的所有向量，依序排列。同一群組的點，

即為一組可能的重複樣式。最後，利用 Heuristic 規則，刪除沒有意義的重複樣式。

Shih 等人[39]修改 LZ78 壓縮演算法中，建立字典(Dictionary)的原理，用以找出音樂旋律以小節為單位的重複樣式。主要的作法有三個步驟。第一，將音樂旋律以小節為單位切割成 Bar pattern。然後，對所有的 Bar pattern 建立索引，並記錄每一個不同的 Bar pattern 的內容、重複出現的次數與一個唯一的編號。利用索引，將原來音樂旋律以索引的編號來表示。第二，以索引編號表示的音樂旋律，透過修改過的 LZ78 演算法，建立出一個字典，而字典內的所有資料即為重複樣式。

2.4.2 探勘趨近重複樣式

Hsu 等人[16]提出 Apriori based 的探勘趨近重複樣式(Mining Approximate Repeating Patterns)方法，找出一首音樂中趨近重複樣式。首先，Hsu 定義了三種不同的 Approximate matching 的方法，分別是 Longer length matching、Shorter length matching 與 Equal length matching。

利用前述的其中一種 Approximate matching 的方法探勘 Approximate 經常重複序列，主要有三個步驟，需給定 Minimum support、容許 Approximate 誤差值(Approximate degree)、序列的長度限制。大致的做法如下，第一，從音樂旋律中找出所有長度為 1 的候選重複序列，並檢驗是否符合 Minimum support，保留符合 Minimum support 的序列。第二，利用前一個步驟產生出來長度為 n 的重複樣式，利用 Join 的方式，組合產生長度為 n+1 的候選重複樣式。第三，利用選定的 Approximate matching 方法，檢驗第二步驟所有候選重複樣式，刪除出現次數低於最低門檻值的樣式。接著重複不斷第二與第三步驟，直到無法產生新的重複樣式或者到達最長。舉例來說，給定一個音樂旋律音高 “A B F C D L B M A B P F C F D”，最低門檻值為 2、容許 Approximate 誤差值為 1、最長樣式長度為 4。所以我們可以從給定的旋律找出下面的重複樣式：P1={ “A”, “B”, “C”, “D”, “F”}，P2={ “AB”, “BF”, “CD”, “FC”, “FD”}，P3={ “ABF”, “BFC”, “FCD”}，P4={ “ABFC”}。

第三章

音樂分段

目前現有的研究中，音樂的分段大都利用人類的聽覺認知(Perceptual)，並不是針對音樂內容做主題式的分段。在一般的音樂裡面，常常包含一個以上的段落。音樂的段落如同文章的段落，在每個段落包含一個主要的音樂主題。沒有研究對音樂做分段的目的，是希望藉著分段後的結果，可以用來分析音樂的高階資訊。我們希望可以針對音樂做主題式的分段，藉此可以讓使用者能夠瀏覽音樂的主題。

音樂的主題段落都包含一個主題，而音樂主題是以音樂動機(Motive)來決定的，作曲者在作曲時，都會以音樂的動機作發展，完成一個主題的段落。因此，動機會以不同的變化形式，重複出現在一個音樂的主題內。我們對整首音樂以探勘重複樣式技術找出動機後，可以再利用動機將主題段落切割出來。

音樂動機序列與傳統的重複樣式特性不相同，主要差別在於樣式比對(Pattern matching)的方法。傳統重複樣式探勘的方法，是以精確比對的方法檢查樣式在序列中重複的次數，當樣式的重複次數超過 Minimum support 時，就是經常重複樣式 (Frequent repeat pattern)。音樂動機不是精確的重複出現，如果直接用傳統比對樣式的方法，並不完全適合。所以我們必須針對動機的特性，設計出適合的方法。音樂的動機包含一組特有的音程與節奏，作曲者在作曲時，利用**動機處理**(Motivic treatment)，不斷的改變動機。音樂學中，有一種音樂分析的方法，稱為曲式分析(Analysis of Musical Form)。針對於音樂動機的部份，音樂學在曲式分析歸納出動機處理的規則。因此，我們可以利用動機處理的規則，對音樂的動機做重複樣式的比對，藉此探勘出音樂的動機。根據 Stein[41]，常見的動機處理有六種，分別為 Repeat、Pitch Interval Repeat (Transposition)、Sequence、Contrary Motion、Retrograde 與 Augmentation 或 Diminution。我們必須針對不同的動機處理，來設計探勘的方法。

我們將找出主題段落的方法分成四個步驟處理如圖 3.1 流程圖所示，並說明如下。

MIDI 檔案必須先透過 MIDI Parser 剖析檔案格式，將 Binary 格式剖析成文字格式資料。剖析後，從文字格式資料中擷取主旋律。我們利用 All-mono 的方法做主旋律擷取。擷

取出來的主旋律會以兩種不同節奏表示法來表示，Original duration representation 與 IOI duration representation。針對兩種表示法，我們接著在下列步驟中分別找出可能的動機。

我們對音樂主旋律先做**粗略分段**(Rough segmentation)，以利於動機探勘。粗略分段的目的是為了幫助動機探勘時，減少誤判的動機產生。在這裡我們以探勘音樂的 Non-trivial 重複樣式，來對主旋律做粗略分段。

主旋律經過粗略分段後，從分段的結果找出動機。我們對產生出來的粗略分段結果，做動機探勘。動機探勘針對動機重複特性修改傳統探勘重複序列的方法，我們將修改後的方法稱為**探勘 Motive Treatment 方法**。

最後，利用找出來的動機對音樂做主題式的**精細分段**(Fine segmentation)。

第一步驟 Parser 輸出的格式與主旋律擷取的方法將在 3.1 介紹；第二步驟的粗略分段作法，詳細的細節會在第 3.2.1 詳加敘述。在第三步驟中，兩種不同探勘動機方法的細節會在 3.3 做進一步的說明。最後一個步驟，我們利用探勘出來的動機作精細分段，我們將在 3.4 做討論。

在進一步說明之前，我們先定義傳統重複樣式。

[定義 3.1]序列 $S = (s_1,s_2,\ldots,s_m)$，存在一個序列 $P=(p_1,p_2,\ldots,p_n)$ 為 S 的**部分序列**(subsequence)，表示成 $P \subseteq S$，如果$\exists\, i, 0 \le i \le m\text{-}n+1$，使得 $p_1=s_i$, $p_2=s_{i+1}\ldots$, $p_n = s_{i+n-1}$ 。反之，我們以 $P \not\subset S$ 表示。

[定義 3.2]對於序列 $P \subseteq S$，存在 $P' \subseteq S$，$p_1=p'_1$, $p_2=p'_2$, …,$p_n=p'_n$。且$\exists\, i, j, 0 \le i, j \le m\text{-}n+1$，$i \neq j$，使得 $p_1=s_i$, $p_2=s_{i+1}\ldots$, $p_n = s_{i+n-1}$，$p'_1=s_j$, $p'_2=s_{j+1}\ldots$, $p'_n = s_{j+n-1}$。我們稱 P 為 S 中的 **Exact 重複樣式**。

[定義 3.3]序列 P 為 S 中一個 Exact 重複樣式，存在一個集合 $PS=\{ ps_1,ps_2,\ldots,ps_k \}$，$\forall ps \in PS$，$0 \le ps \le m\text{-}n+1$，使得 $p_1=s_{ps},p_2=s_{ps+1},\ldots,p_n=s_{ps+n-1}$。k 為 P 在 S 中的 Support，我們以 Support(P)=k 表示。

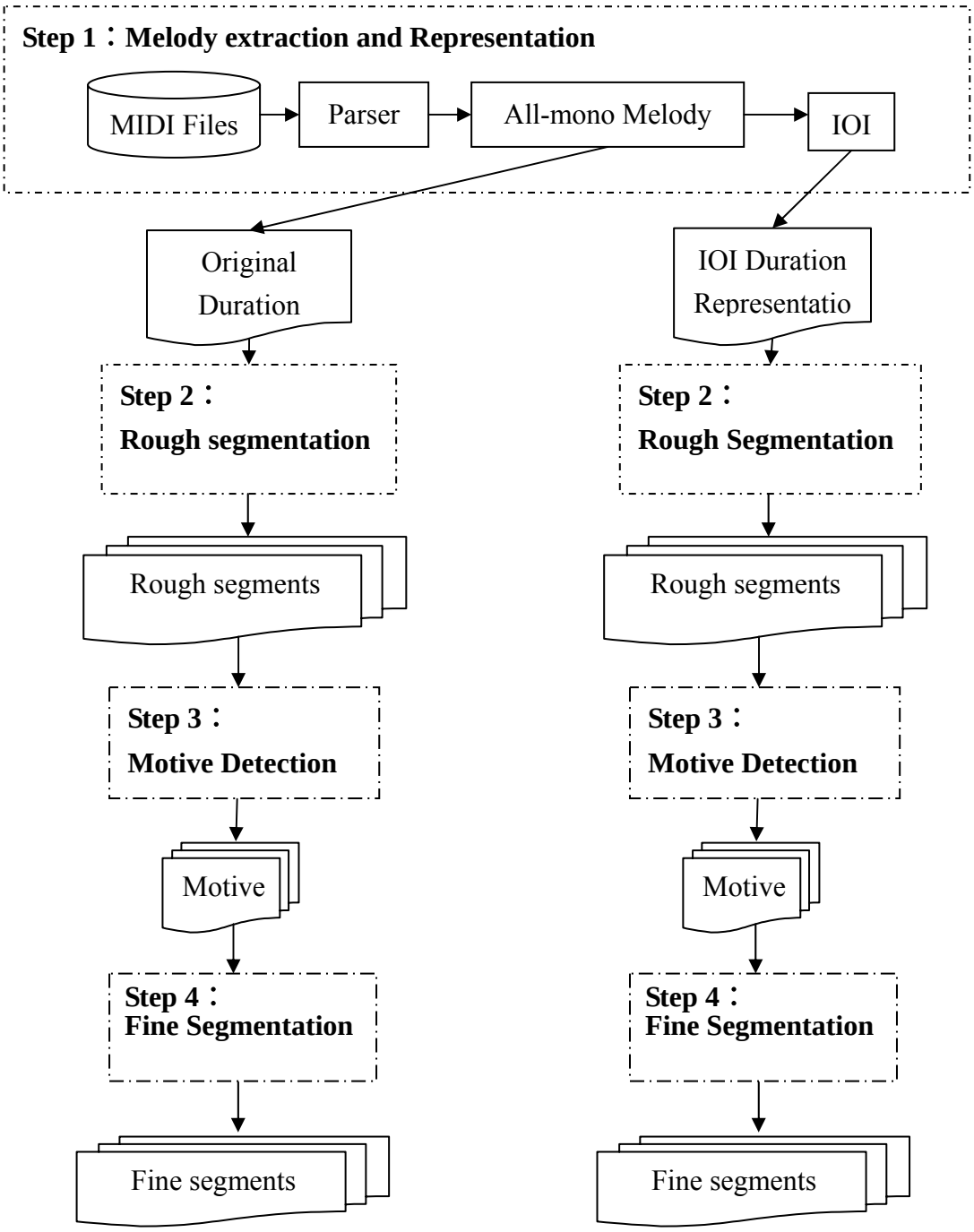


圖 3.1：主題分段流程圖。

[定義 3.4]給定一 Minimum support 值 ms，如果序列 P 在 S 中的支持度 k ≥ ms，則 P 稱之為 S 的**頻繁重複樣式**(Frequent repeat pattern)。

[定義 3.5]當 P ⊆ S，P 為 S 的 Frequent repeat sequence pattern，∀P' ⊆ S，P≠P'，P'為 S 的 Frequent repeat sequence pattern，P ⊄ P'。我們稱 P 為 **Maximal 重複樣式**。

[定義 3.6]當 $P \subseteq S$，P 為 S 的 Frequent repeat sequence pattern，$\forall P' \subseteq S$，$P \neq P'$，P’為 S 的 Frequent repeat sequence pattern，$P \not\subset P'$且 Support(P)≠Support(P’)。我們稱 P 為 **Non-trivial 重複樣式**。

3.1 主旋律的萃取與表示法

MIDI 檔案是以 Binary 的格式儲存資料，在 1.3 中我們已經詳加介紹過 MIDI 的儲存格式。Binary 的格式並不容易直接處理，所以我們利用 MIDI parser 將 MIDI 檔案剖析成其他較容易使用的資料格式。如圖 3.2 所示，我們所使用的 MIDI parser 會將 MIDI 檔案的 Binary 資料轉換成文字型態的資料。

MIDI 檔案並不是直接紀錄音符的資訊，而是紀錄許多個 Events，包含 MIDI 的系統資訊與音符的資訊等。一個音符利用兩種特定的 Events，Onset event 與 Offset event 分別來表示開始演奏與結束演奏。所以，我們將 Events 的資訊，轉換成音符的資訊，以音符的序列表達音樂。我們將 MIDI 檔案轉成以音符表示的序列$<note_1, note_2, \ldots, note_m>$，每個 $note_i$ 表示一個音符，包含有$(onset_i, pitch_i, d_i)$。$onset_i$ 表示音符開始發聲的時間；$pitch_i$ 代表音高；d_i 代表音長。

```
Track 1：
tick 0: Sequence/Track Name: Right Hand
tick 0: channel 2: program change 3
tick 386: channel 2: note On 60 velocity: 127
tick 435: channel 2: note Off 60 velocity: 0
tick 576: channel 2: note On 60 velocity: 127
tick 769: channel 2: note Off 60 velocity: 0
tick 770: channel 2: note On 64 velocity: 127
tick 943: channel 2: note Off 64 velocity: 0
```

圖 3.2：經過 Parser 處理後的 MIDI 資料。

例如圖 3.2 經過 MIDI Parser 處李後的資料，我們以音符表示的結果為<(386, 60, 49), (576, 60, 193), (770, 64, 173) >。針對(386, 60, 49)，圖 3.2 中我們可以找出 tick 386 是音高

60 開始發音的時間，接著 tick 435 是音高 60 結束發音的時間，所以我們可以計算出 60 這個音的音長為 49 個 Tick time。另外兩個音符(576, 60, 193)與 (770, 64, 173)分別是由 tick 576、tick769 與 tick770、tick 943 所決定。圖 3.2 中的兩個 Tick 0 events 與音樂內容沒有直接關係的資訊。

一般音樂包含的內容有主旋律以及伴奏。為了能夠探勘出音樂的主題，我們必須針對主旋律的部分做擷取。因此，我們需要將主旋律從原始音樂資料中擷取出來。而且，音樂的主旋律不完全固定在同一個 MIDI track 中演奏，可能會再多個 MIDI tracks 裡輪流出現。所以我們在轉換 MIDI events 表示法成音符表式的序列時，同時對音樂做主旋律擷取。主旋律是作曲家希望呈現的主題，而伴奏只是為了襯托或修飾主旋律的部分。通常主旋律會比伴奏明顯，因此我們要將音樂中較明顯的音符擷取出來當作主旋律。由於大部分的音樂會同時包含主旋律與伴奏，我們在對音樂主題資料做分析時，不需要去考慮伴奏的部分。

主旋律擷取是參考 Uitdenbogerd [46] 等人擷取主旋律的方法，她們提出了 All-mono、Entropy-channel、Entropy-part 與 Top-channel。All-mono 的方法將一首音樂的所有 MIDI tracks 資料結合成一個，此時會有許多音符有重疊在一起。然後保留同時開始發音中，音高最高的音符作為主旋律。Entropy-channel 方法是計算各別 MIDI track 的 Entropy，保留 Entropy 最高的 MIDI track 當作主旋律。Entropy-part 的方法是將每一個 MIDI track 以固定長度，如：一個小節，切成多個部份，擷取主旋律時，選擇 Entropy 最高的部份。Top-channel 的方法與前面一個方法相似，同樣將每一個 MIDI track 以固定長度切成多個部份，選擇平均音高最高的部份當作主旋律。根據 Uitdenbogerd 實驗的結果，以 All-mono 方法擷取出來主旋律效果最好。所以，我們修改 All-mono 的方法來做主旋律的擷取。

原本 All-mono 的擷取方法是不考慮發聲音符音長，而我們修改成考慮發音的音長。如圖 3.3 所示，a 的部份表示兩個聲部，以原本 All-mono 的方法從這兩個聲部擷取出主旋律，最後會得到 b 的結果。由圖 3.3 第二與第三小節我們可以觀察到，這個方法並沒有考慮一個音符持續發音的情形。舉例來說，a 的兩個聲部中，高音的聲部在第二小節的第一個音符為兩拍的音符，這個音符會持續發音至第二小節的第二個音開始發音為止。All-mono 的方法並不會考慮持續發音的狀況，只考慮同一時開始發音的音符。這樣一來主旋律會被其他伴奏的聲部影響。

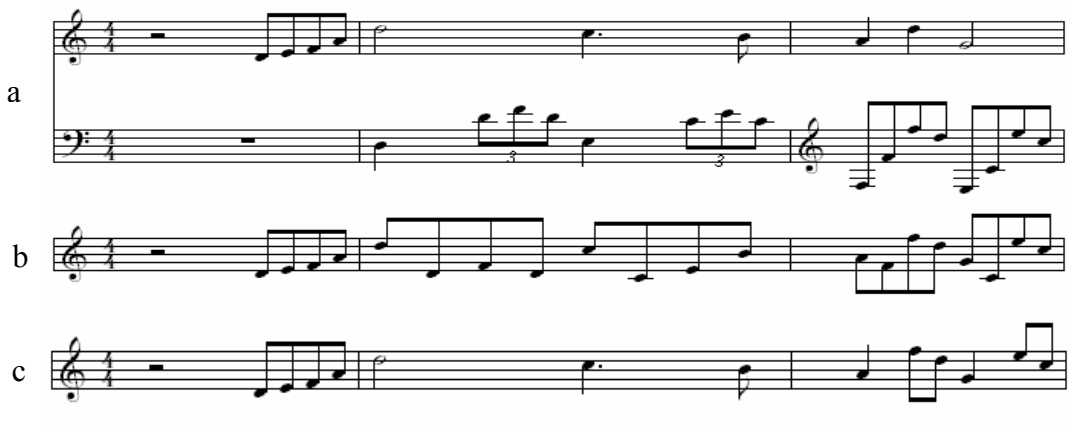


圖 3.3：All-mono 的範例。

我們修改原本的方法加入考慮持續發音的情形，原本 a 的兩個聲部擷取主旋律的結果，會如圖 3.3 中 c 的結果。圖 3.4，長條代表一個音符，虛線處代表新的 Event 發生。一個音樂序列，如果以我們修改的 All-mono 方法做主旋律擷取時，必須考慮虛線上，所有 Event 所發生的改變。經過我們修改的 All-mono 方法，最後找出的主旋律為圖 3.4 中，有顏色的部分。

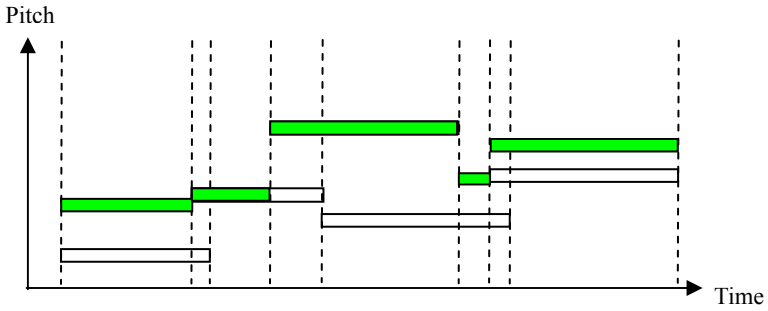


圖 3.4：All-mono 主旋律擷取示意圖。

我們修改後的做法如下。首先，將一首 MIDI 檔案的所有 MIDI Tracks 合成一個 MIDI Track。接著我們依時間順序處理 MIDI Track 中 Event 的資料。只要有新的 Onset event 或 Offset event 發生時，就必須重新檢查目前的最高音，並且記錄下來。如圖 3.4 所示，每一個長條代表一個音符，虛線的地方代表 Onset 或 Offset Events 發生處，最高音的改變都只會在新的 Event 發生處。如此，我們對同一時間的 Event 做處理，也將每一個發音的音長考慮進去。最後，我們可以得到一個序列 $S = <note_1, note_2, \ldots, note_m>$ 表示音樂的主旋律。圖 3.5 是我們所修改的 All-mono 演算法，我們依時間的順序，處理經過 MIDI parser 剖析的資料，

找出屬於主旋律的音符。我們先選擇出目前發音的音符中，最高的音。然後用目前 Onset 與 Offset 的 Events 去設定發音音符的狀態。如果最高的音符已經結束發音或者最高的音符改變，則輸出前面我們所選擇的最高音符到結果的序列中。

擷取出來的主旋律，我們又分別以兩個不同的旋律表達方法做處裡。第一種，用原始音長表示主旋律節奏資訊的 Original duration representation。第二種，用 Inter Onset Interval 的方法表示主旋律節奏資訊的 IOI duration representation。在擷取主旋律的步驟之後，我們計算主旋律中，連續兩個音發生的時間間距。以<(386, 60, 49), (576, 60, 193), (770, 64, 173) > 為例，(386, 60, 49)與(576, 60,193)兩個音的開始發生時間間距為 576 – 386 = 190。所以，我們原本音符的資料(386, 60, 49)，以 IOI duration representation 的表達方法表示為(386, 60, 190)。

3.2 粗略分段

一般傳統的作法在探勘動機時，大都是直接對整首音樂序列探勘音樂的重複樣式。但是，大部分的重複序列可能不是音樂的動機。在一般音樂作曲中，作曲者常常會讓音樂段落直接的重複，會讓音樂段落內的所有子序列都有重複的情形發生。為了減少這種情況，我們可以先針對音樂做**粗略的分段**(Rough segmentation)，產生粗略段落(Rough)再從這些段落中探勘出音樂動機。

```
Algorithm Modified All-mono Melody Extraction
Input: MIDI File E=[e1,e2,…,en]   // each ei contains time, event type, pitch
Output: Melody S
1.  set CurrentTime to e1.time;   set CurrentEventList to empty;
2.  set OnsetList to empty;   // store current onset notes in OnsetList
3.  for each event ei of E do
4.      if ei.time = CurrentTime then AddtoCurrentEventList(ei);
5.      else
6.          set TopPitch = the top pitch of OnsetList;
7.          setOnsetList();
8.          if TopPitch isn't in OnsetList then append TopPitch to S;
9.          set CurrentTime to ei.time;
10.         set CurrentEventList to empty;
11.         AddtoCurrentEventList(ei);
12.     end
13. end
14. set TopPitch = the top pitch of OnsetList;
15. setOnsetList();
16. if TopPitch isn't in OnsetList then append TopPitch to S;
17. return S;
```

```
Function setOnsetList
1.  for each event ei of CurrentEventList do
2.      if ei.type = Offset then remove ei.pitch from OnsetList;
3.      else add ej.pitch to OnsetList; end
4.  end
```

圖 3.5：All-mono 演算法。

如圖 3.6 為例，(a)、(b)與(c)所標示的皆為重複兩次的序列。(a)所標示部分為音樂的動機部分；(b)所標示的部分不是動機；(c)所標示的部分為由動機所構成的主題主要部分，而且(c)為一個 Maximal 的重複樣式，也就是不被其他重複樣式所包含的樣式。由(c)的部分我

們可以觀察到，(a) 與(b)都被包含在內，此外(a)動機也在(c)裡面以 Pitch Interval Repeat 變化的形式出現。由於作曲者會將大部分主題段落重複安排出現，加深聽眾的印象，如圖 3.6 的(c)。所以，我們可以先將這些主要部分用探勘 Maximal 重複樣式的方法先探勘出來，接著對於每個 Maximal 重複樣式再以探勘動機的方法找出動機。我們使用探勘 Non-trivial 重複樣式的方法，來從音樂中探勘粗略段落。

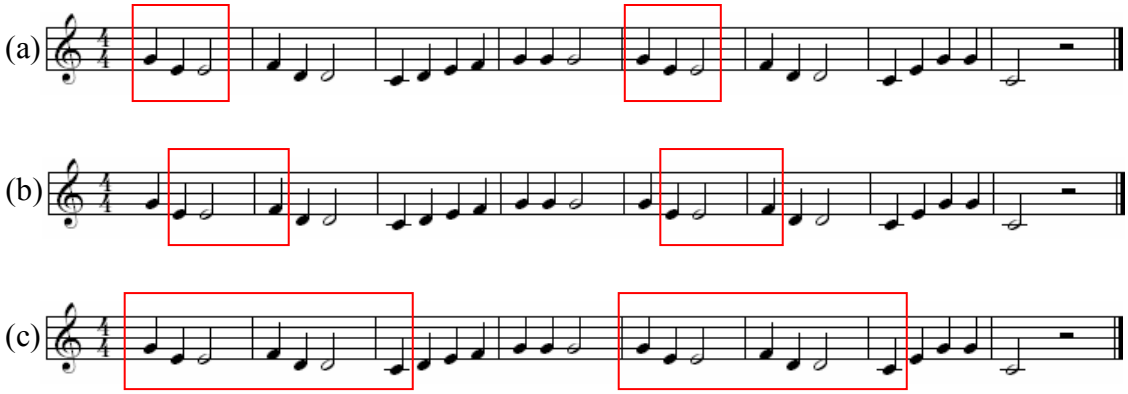


圖 3.6：音樂中的重複樣式。

3.2.1 以 Non-trivail 重複樣式作粗略分段

我們以音樂序列的 Non-Trivial 重複樣式為粗略的分段，我們修改 2.4.1 中 Hsu 所提出探勘頻繁重複樣式的 Correlative matrix 方法。Correlative matrix 做法主要分成三個步驟，第一，建立 Correlative matrix，Correlative matrix 主要在標示同一個序列中，有重複出現樣式的位置，並且紀錄序列的長度。第二，產生候選重複樣式，由於 Correlative matrix 沒有辦法針對所有可能長度的重複樣式做紀錄，所以，必須使用 Correlative matrix 所記錄的資訊，將其他候選序列也產生出來。第三，驗證每一個候選重複樣式是否超過 Minimum support，保留合乎需求的重複樣式。

我們主要修改第二個步驟與第三步驟，以符合找出粗略分段的需求。我們修改第二個步驟中，產生所有可能長度的候選序列的部份，因為我們只需要音符數量超過一定長度的重複樣式，比如說：只有一個音符的 Non-trivial 重複樣式不需要被產生出來，因此我們不需要產生所有的候選重複樣式。第三個步驟，除了檢查是否符合 Minimum support，我們還

增加檢查一個重複樣式的時間長度是否符合我們的需求。以上兩個修改的部份，詳細的方法會再下一章討論。接下來，我們以一個範例來簡單介紹使用這個方法找出音樂的粗略分段。我們先分別介紹第二與第三個步驟原本的作法，然後再個別介紹我們所做的修改。根據第二步驟原來的作法，所有可能長度的重複序列都必須被探勘出來。為了讓探勘出來的重複樣式都是 Non-trivial 的重複樣式，這個方法還會標記重複樣式出現的次數以及被其他重複序列包含的次數。因此，對 Correlative matrix 的每個元素 $D_{i,j}$，有四種不同的處理情形。第一種情形，當 $D_{i,j}=1$ 且 $D_{i+1,j+1}=0$ 時，表示發生一個長度為 1 的重複樣式，而且這個樣式不會被其他樣式包含。第二種情形，當 $D_{i,j} = 1$ 且 $D_{i+1,j+1} \neq 0$，表示發生一個長度為 1 的重複樣式，而且這個樣式被其他樣式所包含。第三種，$D_{i,j} > 0$ 且 $D_{i+1,j+1} \neq 0$，表示發生一個長度為 $D_{i,j}$ 的重複樣式，以及所有的 Suffix 都是重複樣式。而且 $D_{i+1,j+1} \neq 0$，表示所有的重複樣式都會被更長的樣式所包含。第四種，$D_{i,j} > 0$ 且 $D_{i+1,j+1} = 0$，表示發生一個長度為 $D_{i,j}$，而且不會被其他序列所包含的重複樣式，以及所有的 Suffix 都是重複樣式。

```
Algorithm Construct Correlative Matrix
Input: Melody S
Output: Correlative Matrix D
1.  Initialize D
2.  for each si of S do
3.      for each sj of S do
4.          if si = sj then  Di,j = Di-1,j-1 + 1; end
5.      end
6.  end
7.  return D;
```

圖 3.7：建立 Correlative matrix 的演算法[16]。

然而，我們所需要的是 Maximal 的重複樣式。因此，我們在第二步驟的四種情形只要考慮在 Correlative matrix 中，$D_{i,j} > 0$ 而且 $D_{i+1,j+1} = 0$ 這些位置上的重複樣式，以產生較少的候選重複樣式。而第三個步驟，除了原本檢查重複次數是否符合 Minimum support 之外，我們加進重複樣式的長度限制，以過濾掉沒有意義的較短序列。最後，輸出符合這兩個條件的重複序列作為結果。

詳細的演算法如下。首先，第一個步驟，利用音樂主旋律的序列 S，建立一個 Correlative Matrix D。演算法如圖 3.7 所示，對於所有的 si 屬於 S，檢查 sj(j > i)是否為重複出現的音符，並且累計目前連續重複出現的序列長度。第二個步驟，我們只需要找出音符數量大於

一定 Maximal 的重複樣式，對於 Correlative matrix 我們只需要考慮下面的一種情形。當 D 中的一個元素 $D_{i,j} > 0$，而且 $D_{i+1,j+1} = 0$。對應到原來的序列 S，$D_{i,j}$ 指的是一個重複樣式的端點。我們就可以利用 $D_{i,j}$ 的值，倒推出一個重複樣式 P 開始於 S 的第 $i-D_{i,j+1}$ 個位置結束在第 i 個位置。演算法如圖 3.8 所示，我們先把所有符合上述條件的序列找出來，然後找出所有可能的 Maximal 重複樣式，然後再經過檢查重複樣式的長度與出現次數是否符合 Minimum support，輸出最後結果。

```
Algorithm Find Repeating Patterns
Input: Correlative matrix D, shortest length m of one segment
Output: Repeating patterns with constrain
1.  set PatternList to empty
2.  for each d_{i,j} of D do
3.      if d_{i,j} > 0 & d_{i+1,j+1} != 0 then
4.          add all suffix of P = <(s_{i-di,j+1}, s_{i-di,j+2},...,s_i> to PatternList;
5.      end
6.      if d_{i,j} != 0 & d_{i+1,j+1} = 0 then
7.          add P = <(s_{i-di,j+1}, s_{i-di,j+2},...,s_i> to PatternList;
8.      end
9.  end
10. MaximalPatternList = all maximal patterns of PatternList;
11. for each p_i of MaximalPatternList do
12.     if p_i length less than m measures then
13.         remove p_i from MaximalPatternList;
14.     else if frequent of p_i < minimum support then
15.         remove p_i from MaximalPatternList;
16.     end
17. end
18. return MaximalPatternList;
```

圖 3.8：修改的探勘候選 Non-trivial 重複樣式演算法，與檢驗 Non-trivial 重複樣式規則。

[範例 3.1] 以歌曲"小蜜蜂"為例，一個音符以(pi, di)表示，其中 pi 代表音高，di 代表音長。在這裡音高以音樂的音名表示，音長以一拍為單位音長 1。"小蜜蜂"主旋律的序列可以表示為<(67, 1), (64, 1), (64, 2), (65, 1), (62, 1), (62, 2), (60, 1), (62, 1), (64, 1), (65, 1), (67, 1), (67, 1), (67, 2), (67, 1), (64, 1), (64, 2), (65, 1), (62, 1), (62, 2), (60, 1), (64, 1), (67, 1), (67, 1), (60, 2)>。假設探勘重複序列參數 m 的值為 2 個小節，Minimum support 為 2。我們可以建立一個 Correlative matrix 如圖 3.9 所示。在 $D_{i,j} > 0$ 且 $D_{i+1,j+1} = 0$ 為一個重複樣式的結尾，我們可以找出下面幾種重複樣式，<(67, 1)>、<(64, 1)>、<(65, 1)>、<(62, 1)>、<(67, 1), (67, 1)>

與<(67, 1), (64, 1), (64, 2), (65, 1), (62, 1), (62, 2), (60, 1)>等樣式。接著，我們從這幾個樣式中選出 Maximal 的重複樣式，得到<(67, 1), (67, 1)>與<(67, 1), (64, 1), (64, 2), (65, 1), (62, 1), (62, 2), (60, 1)>。此外，我們加入長度的限制，去除總長度短於 2 小節的重複序列，<(67, 1), (67, 1)>會被去除掉。最後一個步驟，我們會檢查重複樣式是否符合 Minimum support 值。最後我們得到一個 Segment 結果<(67, 1), (64, 1), (64, 2), (65, 1), (62, 1), (62, 2), (60, 1)>。

	67,1	64,1	64,2	65,1	62,1	62,2	60,1	62,1	64,1	65,1	67,1	67,1	67,2	67,1	64,1	64,2	65,1	62,1	62,2	60,1	64,1	67,1	67,1	60,2
67,1	–										1	1		1								1	1	
64,1		–							1						2						1			
64,2			–													3								
65,1				–						1							4							
62,1					–			1										5						
62,2						–													6					
60,1							–													7				
62,1								–										1						
64,1									–						1						1			
65,1										–							1							
67,1											–	1		1								1	1	
67,1												–		1								1	2	
67,2													–											
67,1														–								1	1	
64,1															–						1			
64,2																–								
65,1																	–							
62,1																		–						
62,2																			–					
60,1																				–				
64,1																					–			
67,1																						–	1	
67,1																							–	
60,2																								–

圖 3.9：Correlative Matrix 範例。

3.3 動機偵測

我們將介紹兩種從粗略分段的結果中探勘動機的方法。第一種方法，根據 Stein[41]提出作曲者常用的六種動機處理變化特性。我們利用這些特性修改傳統探勘重複序列的方法來探勘動機。第二種方法，利用 Local boundary detection model 的方法，對音樂中所有連續的音符，以數種音樂的特徵值計算出 Boundary strength value，代表兩個音之間落差的大小。

根據計算的結果，選擇 Boundary strength value 高的地方當作分段點。利用這些分段點，將音符群組成許多小段。然後，以群組的小段為單位，找出可能的序列當作動機。

根據 Stein[41]提出常用的六種動機處理變化特性，分別敘述如下。

1. Exact repeating：是動機常見的變化，如圖 3.10 所示。利用反覆的處理，作曲者可以讓聽眾加深對主題動機的記憶。

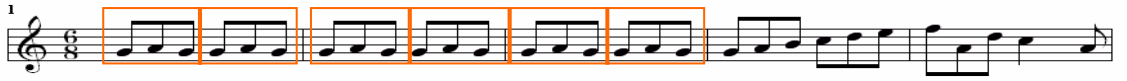
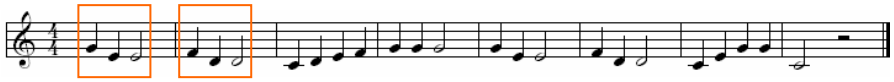


圖 3.10：Exact repeating patterns 的範例。

2. Pitch interval repeating：Pitch interval 是指兩個連續音符的音高差。Pitch interval repeating 在音樂的意義是 Transposition。圖 3.11 為 Pitch interval 重複的範例，純粹看音高時，方框所標示的地方不算重複出現。但是，我們以 Pitch interval 來看，我們就可以知道方框標示的地方是一個重複樣式。



Pitches：|67 64 64|65 62 62|60 62 64 65|67 67 67| ……|
Pitch Interval：|-3 0 1|-3 0 -2|2 2 1|2 0 0|…..|

圖 3.11：Interval repeating patterns 的範例。

3. Sequence(模進)：模進是連續三個以上的動機 Transposition，而且每個 Transposition 不相同，但是必須是相同的升降方向。圖 3.12 為 Sequence 變化的例子，第一個小節內是原始的動機，接著連續兩個 Transposition 變化。

4. Contrary motion(反相)：反相動機處理為保持動機的節奏，但是將旋律反相進行。旋律的反相進行指的是，將動機 Pitch interval 的值，全部取負號。如圖 3.13 所示，第一小節動機的 Interval 為<1, 1, 1, -2, 1, -2>，第二小節的反相變化後的 Tonal interval 為<-1, -1, -1, 2, -1, 2>。

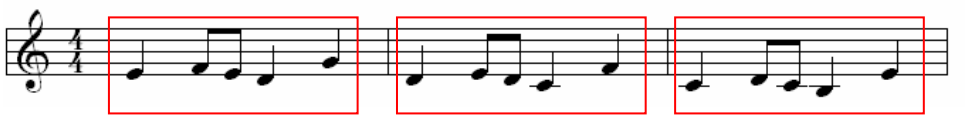


圖 3.12：Sequence 變化的範例。

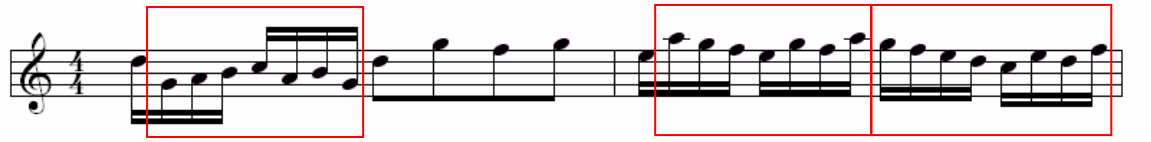


圖 3.13：Contrary Motion 變化的範例。

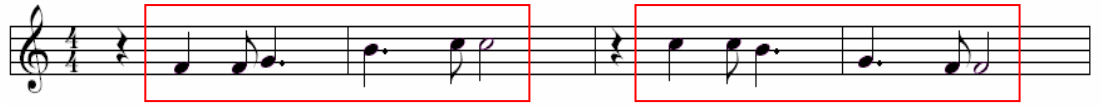


圖 3.14：Retrograde 變化的範例。

5. Retrograde(逆行)：逆行的動機處理是維持原始節奏，將音高倒過來排列。如圖 3.14 所示，第一小節為原始動機，第二小節為逆行的動機變化。整個動機的節奏並沒有改變，音高由原來的<65, 65, 67, 71, 72, 72>變為<72, 72, 71, 67, 65, 65>。

6. Augmentation or Diminution：作曲者在使用增值或減值的動機處理時，會保留原始的旋律，將動機的節奏依比例加快或減慢，也就是延長或縮短動機所有音符的音長。如圖

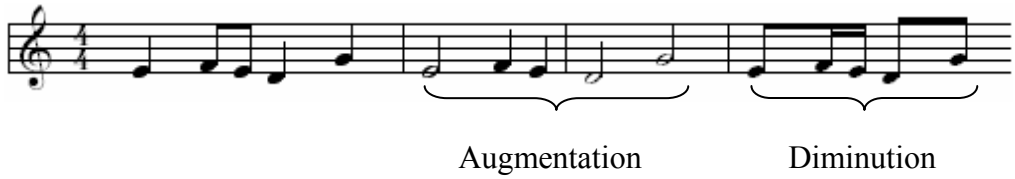


圖 3.15：Augmentation 與 Diminution 變化的範例。

3.15 所示，第一小節為原始動機，第二、三小節為增值的動機，第四小節為減值的動機。

接下來，我們先介紹第一種探勘動機的方法以下我們對探勘動機方法修改的部份做介紹。首先，我們先定義表示音樂特徵資料的符號。

M：代表一段單聲部的動機，記錄有 k 個音符，$k \geq 1$，$|M| = k$；每一個音符記錄一組 (p_i, d_i)，使得 $M = <(p_1, d_1), (p_2, d_2),\ldots, (p_k, d_k)>$。

M_{int}：代表動機 M 的音高部份以 Pitch interval 來表示，每一個音符記錄一組(int_i, d_i)，使得 $M_{int}=<(int_1, d_1), (int_2, d_2),\ldots, (int_{k-1}, d_{k-1})>$，$k \geq 1$，$k = |M|$。

i, j：代表音符在動機中的順序。

p_i：代表一個音符的音高，以半音為單位，根據 MIDI 所制定的規格，$1 \leq p_i \leq 128$。

d_i：代表一個音符的音長或 IOI，以 MIDI 的 Tick time 為計算時間的單位。

int_i：代表前後兩個音的音程差，$int_i = p_i+1 - p_i$。

```
Algorithm Construct Correlative Matrix for Exact repeating
Input: Melody S
Output: Correlative Matrix D
1.  Initialize D
2.  for each si of S do
3.      for each sj of S do
4.          if si = sj then  Di,j = Di-1,j-1 + 1; end
5.      end
6.  end
7.  return D;
```

圖 3.16：Exact repeating 動機變化的 Correlative matrix 演算法。

Exact Repeating

給定一個 $M = <(p_1, d_1), (p_2, d_2),\ldots, (p_k, d_k)>$，利用反覆的規則，可以產生 $M' = <(p_1', d_1'), (p_2', d_2'),\ldots,(p_k', d_k')>$，且 $p_i=p_i'$，$d_i=d_i'$，$0 < i \leq k$。我們利用原始音樂的資料建立 Correlative matrix，然後利用 Correlative matrix 探勘經常重複序列。Correlative matrix 建立的演算法如圖 3.16 所示。在第 3.2.1 節，我們利用 Correlative matrix 的方法找出音樂中 Maximal 的重複序列。所以，我們只考慮 Correlative matrix 中，$D_{i,j} > 0$ 且 $D_{i+1,j+1} = 0$ 的情形。但是，在探勘動機的部分，我們必須對於四種情形全部考慮。

Pitch Interval Repeating

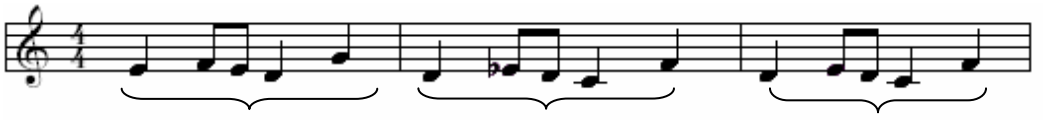
我們直接將音樂轉換成 Pitch interval 的表示方法，探勘出 Interval repeating patterns。給

定一個 M 的 M_{int}=<(p_1, d_1), (p_2, d_2),…, (p_{k-1}, d_{k-1})>，利用反覆的規則，可以產生 M_{int}'=<(int_1',d_1'), (int_2',d_2'),…, (int_{k-1}',d_{k-1}')>，且 int_i=int_i'，d_i=d_i'，$0 < i \leq k$。我們將主旋律產生出來的 Pitch interval 序列，同樣以 Correlative matrix 的方法探勘。我們直接將 Pitch interval 序列使用圖 3.16 的演算法建立 Correlative matrix。

根據 Zinn[50]指出，Transposition 又分為 Real transposition 與 Tonal transposition 兩種。這兩個差異主要源自於 Musical scale 中有 Naturally occurring half steps 所帶來的影響。Real transposition 不需要考慮 Naturally occurring half steps 的影響，只要保持原來的音程差，而 Tonal transposition 必須調整音程差，使其符合 Musical scale 中，Naturally occurring half steps。因此，我們在轉換原始音樂為 Interval 時，分別轉換成 Real interval 與 Tonal interval。

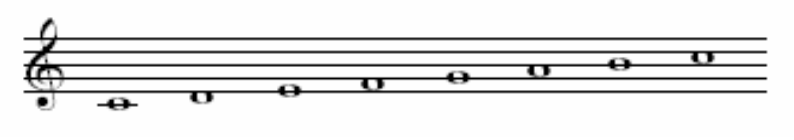
圖 3.17 為兩種 Transposition 的範例，以 MIDI 所記錄的音高來看，第一小節動機的音高序列為<64, 65, 64, 62, 67>；Pitch interval 的序列為<1, -1, -2, 5>。第二小節 Real transposition 的音高序列為<62, 63, 62, 60, 65>；Pitch interval 的序列為<1, -1, -2, 5>。第三小節 Tonal transposition 的音高序列為<62, 64, 62, 60, 65>；Pitch interval 的序列為<2, -2, -2, 5>。

Tonal interval 會受到音樂的 Musical scale 所影響。在音樂中，以 Key signature 標記一個 Musical scale 的起始音。Key Signature 以 Music alphabet 來記錄，如所示。一般我們最常見到的音樂 Key Signature 為 “C”，如圖 3.18 中的 Diatonic Scale 就是以“C” (do)這個音為起始。Key Signature 為“C”的音程為：全音(whole step)、全音、半音(half step)、全音、全音、全音、半音。如果，Key Signature 為“G”(sol)，音程為全音、全音、半音、全音、全音、半音、全音。所以我們會針對不同的 Musical scale 計算主旋律的 Tonal interval。



Original Motive　　Real Transposition　　Tonal Transposition

圖 3.17：兩種 Transposition 的範例。



Syllable：	do	re	mi	fa	sol	la	ti	do
Alphabet：	C	D	E	F	G	A	B	C
Interval：	1	1	1/2	1	1	1	1/2	

圖 3.18：自然音階。

Sequence

Sequence 與前一節提到的 Transposition 一樣會有 Real transposition 與 Tonal Transposition 的情形。給定一個動機 M 的 $M_{int}=<(int_1, d_1),(int_2, d_2),\ldots,(int_{k-1}, d_{k-1})>$，利用 Sequence 的動機變化規則產生 $M_1, M_2,\ldots,M_L$ 連續的 Ascending 或 descending 序列，$L \geq 2$。對於所有 $1 \leq l \leq L$，$M_{lint}=<(int_{l1}, d_{l1}),(int_{l2}, d_{l2}),\ldots,(int_{lk-1}, d_{lk-1})>$，且 $int_i = int_{li}, d_i = d_{li}$，$1 \leq i < k$。建立 Correlative matrix 的方法同於 Pitch interval repeating。

	1,60	-1,30	-1,30	3,60	-3,60	1,60	-1,30	-1,30	3,60	-3,60	1,60	-1,30	-1,30	3,60
1,60	–					1					1			
-1,30		–	1				2	1				2	1	
-1,30			–				1	3				1	3	
3,60				–					4					4
-3,60					–					5				
1,60						–					6			
-1,30							–	1				7	1	
-1,30								–				1	8	
3,60									–					9
-3,60										–				
1,60											–			
-1,30												–	1	
-1,30													–	
3,60														–

圖 3.19：針對 Sequence 變化所建立的 Correlative matrix。

圖 3.12 整首音樂如果以 Tonal interval 的表示法，60 ticks 為 1 拍，30 ticks 為半拍，可

以表示為< (1, 60), (-1, 30), (-1, 30), (3, 60), (-3,60), (1, 60), (-1, 30), (-1, 30), (3, 60), (-3, 60), (1, 60), (-1, 30), (-1, 30), (3, 60) >。由這個序列可以建立如圖 3.19 的 Correlative Matrix，建立的方法與前面探勘頻繁重複樣式的方法相同。接著，利用 Correlative matrix 的資料來產生符合 Sequence 動機變化的序列。

產生序列的方式與小節的方法相同，找出所有 Exact repeat 的序列。接著，我們必須針對每個序列檢查是否有連續出現的情形。由於前一個步驟產生 Exact repeat 的序列時，我們會記錄對應原始序列的位置。所以，我們直接檢查序列的發生位置是否有符合連續的情形，將符合 Sequence 特性的序列輸出。

Contrary Motion

給定動機 M 的 M_{int}=<(int_1, d_1), (int_2, d_2),…, (int_{k-1}, d_{k-1})>，利用 Contrary Motion 可以產生出 M'=<(int_1', d_1'), (int_2', d_2'),…, (int_{k-1}', d_{k-1}')>，其中 int_i=-int_i'，d_i=d_i'，$0 < i \leq k$-1。為了能夠找出 Contrary Motion 的 Repeating patterns，我們修改 Correlative Matrix 建立的方法。由於 Correlative Matrix 是記錄序列資料有重複的部份，所以我們利用原始序列與反相處理過的序列來建立 Correlative Matrix，如圖 3.20 所示。

```
Algorithm Construct Correlative Matrix for Contrary motion
Input: Melody S
Output: Correlative Matrix D
1.  Initialize D
2.  S'=contrary of S;
3.  for each s_i of S do
4.      for each s_j' of S' do
5.          if s_i = s_j' then  D_i,j = D_i-1,j-1 + 1; end
6.      end
7.  end
8.  return D;
```

圖 3.20：Contrary motion 動機變化的 Correlative matrix 演算法。

圖 3.13 前二十個音符的 Tonal interval 序列，以 60 ticks 為一拍，可以表示成<(-4, 15), (1, 15), (1, 15), (1, 15), (-2, 15), (1, 15), (-2, 15), (4, 15), (3, 30), (-1, 30), (1, 30), (-2, 30), (3, 15), (-1, 15), (-1, 15), (-1, 15), (2, 15), (-1, 15), (2, 15)>與反相處理的序列<(4, 15), (-1, 15), (-1, 15), (-1,

15), (2, 15), (-1, 15), (2, 15), (-4, 15), (-3, 30), (1, 30), (-1, 30), (2, 30), (-3, 15), (1, 15), (1, 15), (1, 15), (-2, 15), (1, 15), (-2, 15)>。圖 3.21 為利用前述兩個序列所建立的 Correlative Matrix，我們同樣可以利用這個 Correlative Matrix 找出反相序列的樣式。

	-4,15	1,15	1,15	1,15	-2,15	1,15	-2,15	4,15	3,30	-1,30	1,30	-2,30	3,15	-1,15	-1,15	-1,15	2,15	-1,15	2,15
4,15	-							1											
-1,15		-												1	1	1		1	
-1,15			-											1	2	2		1	
-1,15				-										1	2	3		1	
2,15					-												4		2
-1,15						-								1	1	1		5	
2,15							-										2		6
-4,15								-											
-3,30									-										
1,30										-	1								
-1,30											-								
2,30												-							
-3,15													-						
1,15														-					
1,15															-				
1,15																-			
-2,15																	-		
1,15																		-	
-2,15																			-

圖 3.21：針對 Contrary motion 變化所建立的 Correlative matrix。

Retrograde

給定一個 $M=<(p_1, d_1),(p_2, d_2),\ldots,(p_k, d_k)>$，利用逆行的規則，可以產生 $M'=<(p_1', d_1'),(p_2', d_2'),\ldots,(p_k', d_k')>$，且 $p_i=p_{k-i+1}'$，$d_i=d_i'$，$0< i \leq k$。同樣的，我們修改 Correlative matrix 的建立方法，來探勘出逆行的動機變化，如圖 3.22 演算法所示。

```
Algorithm Construct Correlative Matrix for Retrograde
Input: Melody S
Output: Correlative Matrix D
1.  Initialize D
2.  S'=retrograde of S;
3.  for each si of S do
4.      for each si' of S' do
5.          if si = si' then  Di,j = Di-1,j-1 + 1; end
6.      end
7.  end
8.  return D;
```

圖 3.22：Retrograde 動機變化的 Correlative matrix 演算法。

	65,60	65,30	67,90	71,90	72,30	72,120	72,60	72,30	71,90	67,90	65,30	65,120
65,60	-	1									1	1
65,30		-									2	1
67,90			-							3		
71,90				-					4			
72,30					-	1	1	5				
72,120						-	6	1				
72,60							-	1				
72,30								-				
71,90									-			
67,90										-		
65,30											-	
65,120												-

圖 3.23：針對 Retrograde 變化所建立的 Correlative martrix。

我們以相反的方向來計算 Correlative matrix，如圖 3.23，以 60 ticks 為一拍。原來建立 Correlative Matrix 時，$D_{i,j}$ 會參考 $D_{i-1,j-1}$ 的值做累加，在處理 Retrograde 時，每個 $D_{i,j}$ 需要參考 $D_{i+1,j+1}$ 來做累加。利用建立出來的 Correlative matrix 可以探勘出 Retrograde 的動機變化。在原本方法中，$D_{i,j}$ 有四個不同情形的處理。在這裡我們修改原本建立 Correlative matrix 的方法後，原本的 $D_{i,j}$ 條件必須做改變。所以，我們將第一種情形的條件改為 $D_{i,j} = 1$ 且 $D_{i-1,j+1} = 0$；第二種情形改為 $D_{i,j} = 1$ 且 $D_{i-1,j+1} \neq 0$；第三種情形改為 $D_{i,j} > 1$ 且 $D_{i-1,j+1} \neq 0$；最後一種改為 $D_{i,j} > 1$ 且 $D_{i-1,j+1} = 0$。

Augmentation or Diminution

給定動機 M 的 $M_{int}=<(p_1, d_1),(p_2, d_2),\ldots, (p_k, d_k)>$，利用增減值的處理可以得 $M'=<(p_1', d_1'), (p_2', d_2'),\ldots,(p_k', d_k')>$，而且 $p_i = p_i'$，$d_i = m \cdot d_i'$，$k \geq 1$，$m > 0$。由於增、減值的動機變化不會改變動機的旋律，只會改變動機的音長。所以我們在建立 Correlative matrix 時，純粹考慮音高的資訊來建立。接下來，產生重複序列時，我們會檢查序列音長是否有以相同的比例增長或縮短。從 Correlative matrix 中，我們可以透過座標得知序列比對成功的位置。所以，我們可以計算比對的序列與被比對的序列，兩者的音長是否呈固定的比例。

	64,60	65,30	64,30	62,60	67,60	64,120	65,60	64,60	62,120	67,120	64,30	65,15	64,15	62,30	67,30
64,60	-		1			1		1			1		1		
65,30		-					2					2			
64,30			-			1		3			1		3		
62.60				-					4					4	
67,60					-					5					5
64,120						-		1			6		1		
65,60							-					7			
64,60								-			1		8		
62,120									-					9	
67,120										-					10
64,30											-		1		
65,15												-			
64,15													-		
62,30														-	
67,30															-

圖 3.24：針對 Augmentation 或 Diminution 變化所建立的 Correlative martrix。

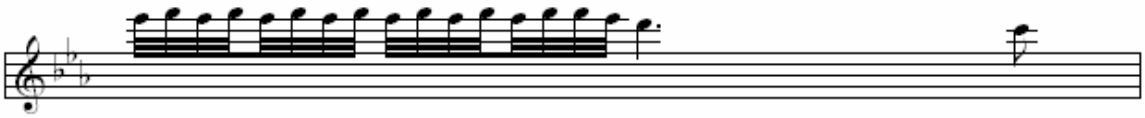


圖 3.25：兩個音快速的交替演奏。

Integration of Motives

我們分別利用上述六種方法探勘出動機後，必須將所有的結果整合使用。我們將序列表示方法，轉換回原始序列以音高與音長表示的資料。接著，從所有動機的序列中，找出 Maximal 的重複序列。最後保留下來的重複序列，我們分別記錄各個動機以不同動機變化出現的次數，以及紀錄一個序列總音長。

此外，我們會將一種由兩個不同音高的音符，快速的交替彈奏的序列刪除，如圖 3.25 所示。這種序列在音樂中，通常是扮演裝飾的角色，與主題無關。

3.4 精確分段

前面我們將音樂作粗略的分段後，從這些粗略段落中，再探勘出音樂的動機。接下來，

我們會利用動機與粗略分段的結果找出精確的段落。根據動機探勘的結果，我們保留存在有動機的粗略段落，將完全沒有探勘出動機的粗略段落刪除。此外，保留下來含有動機的粗略段落，我們會進一步做合併的動作。接下來，我們介紹決定 Fine segmentation 的幾個步驟。

首先，刪除不包含任何動機的粗略分段。第二，我們利用動機以及粗略段落的結果，對粗略段落做 Split 與 Merge，找出同一個動機所構成的段落，我們稱之為 Theme segments。 第三，音樂中除了 Theme segments 以外剩下的段落，我們稱之為 Non-theme segments。我們將長度超過一定量的 Non-theme segment，也定成一個 Theme segment。

在第一個步驟中，我們會刪除沒有包含動機的粗略段落。在探勘動機時，我們會紀錄各個段落探勘出來的動機。如果一個粗略段落無法探勘出任何的動機，很可能是因為該粗略段落是屬於伴奏或間奏較為不重要的部份或者沒有一個明顯的主題，所以我們無法從中探勘出重複出現的動機序列。因此，我們會刪除不包含動機的段落。

第二個步驟，我們檢查動機在各個粗略段落分佈的情形。首先，我們先介紹將粗略段落做 Split 的情形。如圖 3.26 所示，每一個長條代表一個粗略段落，長條中顏色所標的的部分為動機或動機變化發生的位置，不同顏色代表不同的動機。圖 3.26 上方的粗略段落中，包含兩種不同的動機分佈在其中，相同的動機是連續在一起，中間沒有出現其他動機。在這個情形下，我們將粗略段落做 Split，Split 的切割點在一種動機開始出現的位置上。圖 3.26 下方的粗略段落中，包含數個不同的動機，而且分散在其中，沒有連續的出現。對於這種情形，我們不會做 Split，而將整個粗略段落當作可能的 Theme segment。接者我們介紹將粗略段落做 Merge 的情形。如圖 3.27 所示，一個長條代表一首完整的音樂，長條中有顏色所標示的部分代表不同的粗略段落，相同顏色的粗略段落代表由相同的動機所構成。圖 3.27 上方的長條中，第一、二、四與五個粗略段落由相同的動機所構成，第三與第六個粗略段落是由同一個動機所構成。我們檢查相鄰在一定範圍內的兩個粗略段落是否由同一個動機所構成，如果由相同的動機所構成，我們會將兩個粗略段落合併為一個，如圖 3.27 下方長條所示。

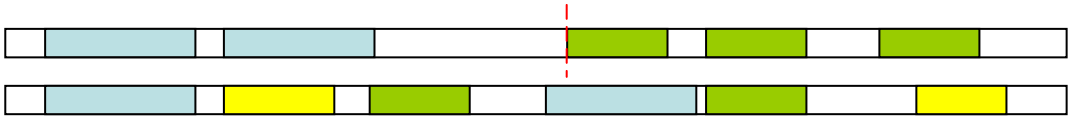


圖 3.26：兩種動機在粗略段落中分佈的情形。

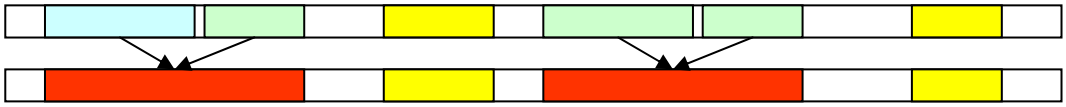


圖 3.27：合併相同動機的粗略段落。

第四章

系統實作

本章介紹系統實做的相關議題與解決的辦法。音樂內容所包含的特徵值非常多，針對不同的需求，我們需要擷取不同的特徵值以供處理。因此，我們必須擷取適合探勘段落或動機的特徵值。舉例來說，我們探勘音樂動機的步驟中，利用音樂動機處理的特性。音樂動機處理中，會利用音樂的 Pitch interval 特徵。Pitch interval 的特徵表示法又分為兩種，一種稱為 Real interval，另外一種稱為 Tonal interval。Tonal interval 會受到 Musical Scale 所影響，我們必須要考慮因為不同的 Musical Scale 所造成的 Tonal interval 變化，這些變化會造成結果的不同，詳細差別細節會在後面的章節討論。

除了 Pitch interval 外，節奏的資訊也因為特性上的不同，必須多做考慮。根據根據 Dixon[13]等人所述，大部分節奏的資訊主要是音符開始演奏的時間，而 Inter-Onset Interval(IOI)的表示法較為符合這個特性。舉例來說，一般的音樂盒所演奏的音樂，並不會按照原來音樂的音長來發聲。因為音樂盒的每個音都只有在開始發聲的時候被彈奏，並沒有理會一個音的長短是否不足或過長。但是，我們人類卻依然可以辨識出音樂和所播放的音樂。因此，我們可以使用 IOI 表示節奏，相對於原始音長，讓資料更適合於音樂的動機探勘。

4.1 調性音階

由於，音樂的音高關係會因為不同的 Musical scale 而有不同音程差關係。所以，我們將 MIDI 檔案原本所記錄的音高，依照對應的 Musical scale 轉換成 Tonal pitch 的表示法。如圖 4.1 上方所示，C 大調的音階實際在 MIDI 檔案所記錄的資訊是<60, 62, 64, 65, 67, 69, 71, 72>；而下方的 G 大調音階實際在 MIDI 檔案所紀錄的是<60, 62, 64, 66, 67, 69, 71, 72>。如果考慮 Tonal transposition 的情形，在 C 大調的 64 與 65 應該視為一個間距一個音，而在 G 大調中的 64 與 65 視為半個音。所以，針對 Musical scale 的不同，我們必須做不同的對

應。在 MIDI 檔案格式中，會紀錄一個 Key signature。Key signature 在音樂中，決定 Musical scale。由此我們可以將 MIDI 原本的音高資訊，對應回 MIDI 檔案所設定的 Musical scale。

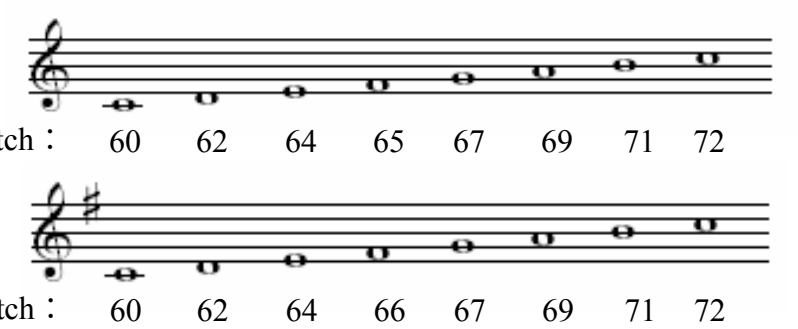


圖 4.1；C 大調與 G 大調音階。

我們建立一個對應原始音高與 Musical scale 的表格，將各個不同的 Musical scale 對應於原始 MIDI 音高的資訊記錄在表格內。透過我們對應的表示方法，C 大調音階的 MIDI 音高<60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72>，經過處理後，原來的序列會變成<36, 36.5, 37, 37.5, 38, 39, 39.5, 40, 40.5, 41, 41.5, 42, 43>。其中<61, 63, 66, 68, 70>是音階中的升降音，在轉換後會變為<36.5, 37.5, 39.5, 40.5, 41.5>。

4.2 音長區間量化

MIDI 是紀錄製作者演奏的資訊，在彈奏時，常常會有誤差。所以在 MIDI 檔案紀錄的拍子長度，通常都是不準確的。而我們比對音符時，必須同時考慮音高與音長。因此，我們必須針對音長的部分做 Quantization。在 MIDI 檔案中，記錄有 Time signature 的資訊，這個 Event 會紀錄 1 拍的長度是多少個 tick time。利用這項資訊，我們可以矯正 MIDI 製作過程所發生的誤差。針對音長長短不同，我們使用不同的解析度來矯正音長。當音長為一拍以上時，我們以 1/2 拍做 Quantization；如果小於一拍，我們以 1/4 拍做 Quantization。

4.3 動機整合

我們在探勘動機時，分別對音高與音長使用不同的表示方法，以利我們使用動機處理的特性。為了讓我們整合多種不同表示法所代表的動機，我們會把所有探勘出來的動機，

對應找出原始序列的資料。為了有效率的找出對應原始序列，我們紀錄探勘出來動機的 Onset time，讓我們加快找出對應的序列。因此，所有的表示法我們都可以透過 Onset 的資訊找出對應的原始序列。

4.4 MIDI parser

MIDI 檔案有許多種不同的格式，主要分為 GM、GS 與 XG 三種。第一種 GM 格式是一般標準的 MIDI 格式，也是最常見的格式。我們實驗測試的資料以 GM 格式為主。第二種 GS 格式是由 Roland 公司所提出，提供比 GM 多的音色與特殊的 Events，用來做更複雜的作曲工作。最後一種 XG 是由 YAMAHA 公司所提出，與 GS 的目標相同，一樣提供更多不同的音色與特殊的 Events。這三種不同的格式，其 Event 儲存的方式也不太相同。我們利用一個由 Matthias Pfisterer 所製作的 MIDI parser 來剖析 MIDI 檔案的 Binary 格式。這個 Parser 是以 Java 提供 MIDI 的類別(class)所製作，主要能將 GM 的 Binary MIDI 格式，轉換成文字表示的 Events。我們保留需要的 Event 資訊，包含有下列兩種。第一種，與整首音樂有關的 Events。Time signature，標示音樂中，每一個小節的拍長。Resolution，標示一拍的音長有多少個 Tick time。Key signature，標示音樂所使用的 Musical scale。第二種，與音符有關的 Events。Note On，表示一個音符開始發生的 Event，以及 Note Off，表示一個音符結束發音的 Event，兩者都包含有音高與音量的資訊。

4.5 轉換 MIDI 格式

我們對 MIDI 檔案作分段或動機探勘後，必須將我們找出來的重複序列轉換回 MIDI 的檔案格式，以供使用者可以在網頁上收聽。因此，我們使用一個可以以文字資料轉換成 MIDI 檔案的工具。由於我們是以音符表示方式來探勘段落與動機等資訊，所以我們利用一個由 Guenter Nagler 所提供的工具“Dmp2midi”。我們先將探勘出來的結果，轉成 Dmp2midi 所使用的輸入格式，如圖 4.2 所示。然後，資料輸入到 Dmp2midi 中轉換成 MIDI 檔案。Dmp2midi 的輸入格式與一般五線譜記譜的方式相似，與 MIDI 的 Event 不同。Dmp2midi 以音符為單位，紀錄每一個音符開始發生的時間、音高、音量與音長。圖 4.2 表示七個音的資訊，第一小節有三個音，第二小節有四個音。

```
192 units per quarternote
Track 1:
M001      ======= Beat: 4/4
M001      1.000     C6 V100     2.000
M001      3.000     B5 V100     1.096
M001      4.096     F5 V100     0.096
M002      ======= Beat: 4/4
M002      1.000     A5 V100     1.000
M002      2.000     G5 V100     1.000
M002      3.000     F5 V100     1.000
M002      4.000     D5 V100     1.000
```

圖 4.2：Dmp2midi 的文字格式。

4.6 系統介面

我們建立一個簡單的系統，簡單介紹各個步驟，系統的介面。圖 4.3 為系統首頁，可以讓使用者聆聽原始的 MIDI。經由粗略分段的結果，將每一段以 Motivic treatment 方法找出的動機，我們分別統計出各個動機變化出現的數量，以及動機的長度。圖 4.4 為最後精細分段結果呈現的頁面，所有段落的長度也會被呈現出來。四個步驟的 MIDI 音樂檔都可以直接在線上收聽。

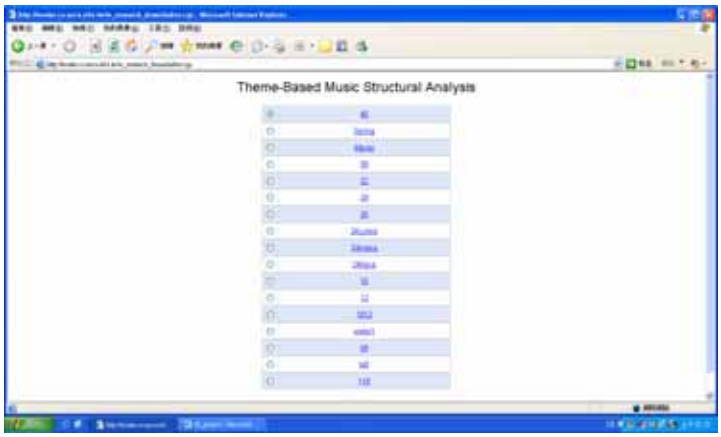


圖 4.3：原始 MIDI 檔案頁面。

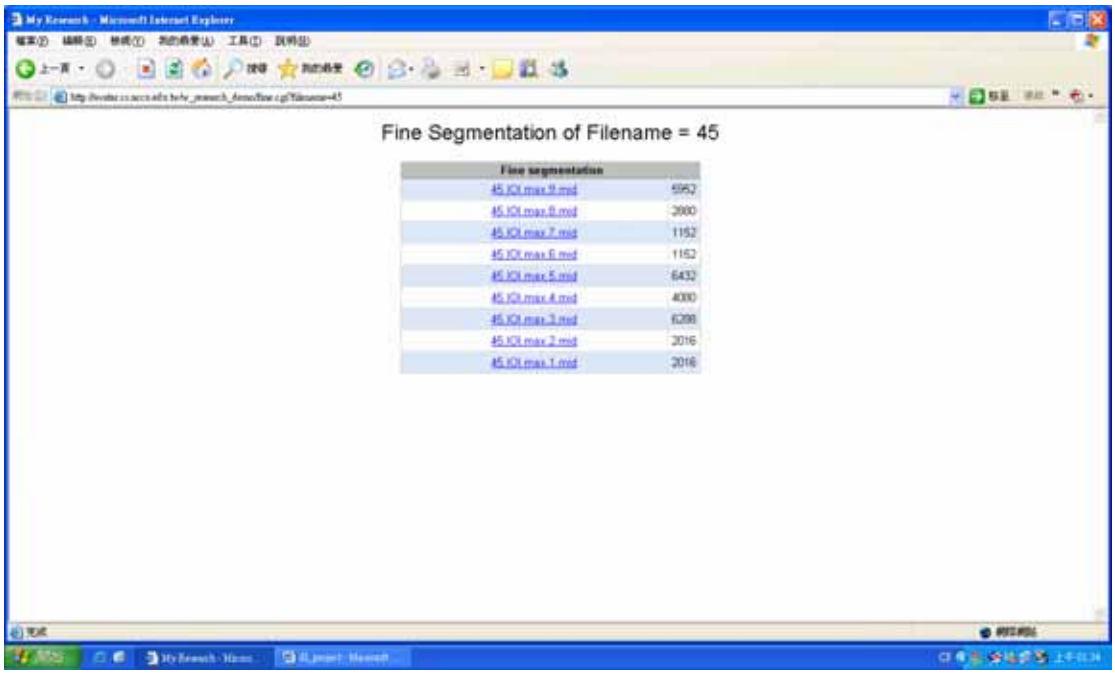


圖 4.4：最後找出精細段落的頁面。

第五章

實驗

5.1 實驗資料與實驗設計

本研究目的在於提供使用者可以瀏覽音樂的主題段落。所以我們針對 10 首不同的音樂，做主題分段。表 5.1 為我們所實驗的音樂列表，包含有巴洛克時期、古典時期、浪漫時期等不同時期。所有的音樂我們請兩位受過專業音樂教育的愛樂者，以主題為基礎做分段，以作為評估實驗結果的 Ground truth。

表 5.1：音樂資料列表

作曲者	曲名	時間長度
海頓	小夜曲	05:43
莫札特	土耳其進行曲	03:44
柴可夫斯基	胡桃鉗序曲	03:18
芭達潔芙絲卡	少女的祈禱	03:52
布拉姆斯	圓舞曲	01:42
莫札特	小步舞曲	02:26
貝多芬	給愛麗斯	02:56
巴哈	聖母頌	02:25
布拉姆斯	匈牙利舞曲	02:03
韋瓦第	春之頌	03:08

在我們所提出來的方法中的第一步驟對於所有音樂，我們針對音長的部分產生兩份不同的資料，分別以 Original Duration 與 IOI 的表示法來表示。接者分別從兩份資料找出粗略分段結果、動機以及最後的精細分段結果。最後，我們將兩份個別的精細分段結果與 Ground truth 做比較，評估我們所提出的方法效果。

5.2 實驗評估方法

目前的相關研究在評估準確率時，常用 Precision 以及 Recall 兩種評估值。Precision 的值表示找出來的分段結果中，符合 Ground truth 的比例；Recall 的值表示找出來的分段結果

包含的 Ground truth，與所有 Ground truth 的比例。至於，判斷是否與 Ground Truth 相符合，是比較找出來的一個分段與 Ground truth 對應的段落，是否誤差在一定的範圍內。

然而，考慮 Precision 與 Recall 可能造成兩種問題的發生。第一種，當我們將一段 Ground truth 切成一個以上的段落時，也就是多找出幾個分段點，所有的結果都會被判斷為不符合 Ground truth，造成 Precision 或 Recall 的值降低。如圖 5.1 所示，下方的長條圖為分段結果的 Gruond truth，上方為欲評估的分段結果。長條圖內的數字代表小節。假設誤差不能超過 3 個小節，我們在第一段的地方多分出了一段，因為超過誤差範圍，所以在計算 Precision 和 Recall 時，會把第一段的結果全部當作不符合。第二種，依據我們的方法所切割出來的結果與 Ground truth 都相近，但是都超過誤差時，會被判斷為全部不符合使得 Precision 與 Recall 的值為 0。如圖 5.2 所示，上方為 Ground truth 的結果，下方為透過我們方法分段的結果，由於每一個小節都超過誤差，所全部都不能算是符合。因此，Precision 與 Recall 並不能表示出相似的程度，所以我們必須利用其他能夠呈現相似度的評估方法來對結果作評估。

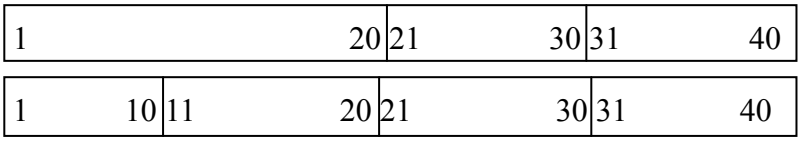


圖 5.1：Precision 與 Recall 誤判的第一種情形。

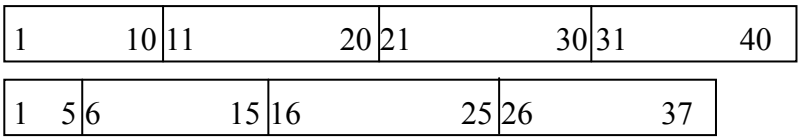


圖 5.2：Precision 與 Recall 誤判的第二種情形。

我們設計一個新的評估實驗效果方法，比較我們方法所找出來的結果以及 Ground truth 之間的相似度，我們令 U=<u1, u2,…, um>代表我們方法的分段結果，V=<v1, v2,…, vn>為 Ground truth。ui 或 vj 以(starti,endj)或(startj,endj)所表示，start 表示段落 ui 開始的小節，end 表示段落 ui 結束的小節。

[範例 5.1] 一首音樂段落的情形為<(1,5), (7,11), (15,20)>，表示這首音樂總共分為三段，第一段由第一個小節開始，到第五個小節結束；第二段由第七個小節開始，第十一個小節結

束；第三段由第十五個小節開始，第二十個小節結束。

接者我們介紹評估方法相關的幾個定義：

[定義 5.1] 給定 U=<u1, u2,…, um>與 V=<v1, v2,…, vn>，U 與 V 之間的對應(Mapping)Rm 是一個一對一的關係。|Rm| = min{m, n}；且在 Rm 中任兩個(ui, vj)與(uk, vl)，如果 i < k，則 j < l。

[定義 5.2] 給定 U=<u1, u2,…, um>與 V=<v1, v2,…, vn>，U 與 V 依據 Rm 的對應關係，所表示的相似度為 $S'_{R_m} = \sum_{\forall(i,j)\in R_m} s(u_i, v_j)$ 。其中 s(ui,vj)代表 ui 與 vj 間的相似度。

[定義 5.3] 相似度 s(ui, vj)的計算方法，是計算兩者共同擁有的小節數量，並且利用此兩段落的小節數量做 normalize。計算公式如下：

$$\text{segment similarity} = \frac{\text{number of overlaped measures of } u_i \text{ and } v_j}{\sqrt{\text{length of } u_i}\sqrt{\text{length of } v_j}}$$

[範例 5.2] 假設我們要比較的兩個段落，分別是 u_i = (8,43)與 v_j = (5,29)。我們可以找出共同有的小節為 8 到 29 共 22 個小節，u_i 的長度為 36 個小節，v_j 的長度為 25 個小節。所以我們可以計算出 u_i 與 v_j 兩個段落的相似度為 $\frac{22}{\sqrt{36}\sqrt{25}}$ 。

[定義 5.4]給定 U=<u_1, u_2,…, u_m>與 V=<v_1, v_2,…, v_n>，U 與 V 之間的相似度 S(U,V)定義為 $\max_{\forall R_m}\{S_{R_m}(U,V)\}$ 。

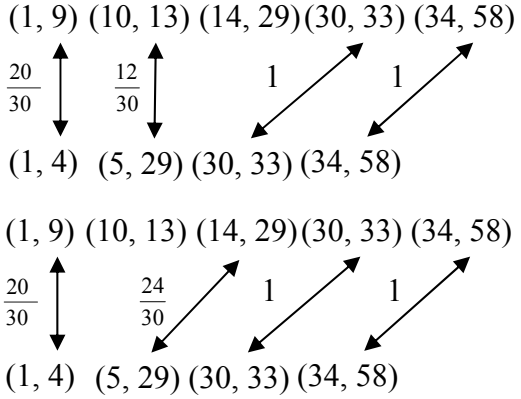


圖 5.3：兩種序列計算相似度對應的方法。

對於整個 U 與 V 來說，由於 U 與 V 可能有很多不同的對應方式，計算對應的段落相似度時，也會有好壞不同的結果。舉例來說，U = <(1,9), (10,13), (14,29), (30,33), (34,58)>，

V=<(1,4), (5,29), (30,33), (34,58)>，可能有圖 5.3 所示的兩種對應方法。圖 5.3 的上方，是將 U_2 對應到 V_2 的情形，下方是將 U_3 對應到 V_2 的情形，兩者在總體所計算出來的相似度並不相同。因此，我們必須考慮所有對應的情形，並且挑出最佳的結果。

我們可以應用 Dynamic programming 的技術建立一個矩陣 S，求出 S(U,V)的結果。在 Dynamic programming 的計算過程中，我們只考慮下面兩種情形。第一種為 Map 的情形，對於 S[m, n] = S[m-1, n-1] + $s(u_m,v_n)$。第二種為 Ignore 的情形，對於 S[m, n] = S[m, n-1]。所以我們可以歸納出下面建構 S 矩陣的公式：

$$S[m,n] = \max\begin{cases} S[m-1,n-1] + s(u_m, v_n) \\ S[m,n-1] \end{cases}$$

對於所有 $1 \le j \le n$，S[0, j] = 0。對於所有 $1 \le i \le m$，S[i, 0] = 0。圖 五.4 是利用上面所定義的方法，計算 S(U, V)的演算法。

我們以 U = <(1,9), (10,13), (14,29), (30,33), (34,58)>，V=<(1,4), (5,29), (30,33), (34,58)>為例，我們可以建構出如圖 5.5 的矩陣。其中，圓圈所標記的部分代表 U 與 V 之間的對應關係。所以我們可以求得 U 與 V 的相似度(Similiarty)為 104/30。最後，我們必須以長度作 Normalize，得到最後相似度的結果，所有的結果會介於 0 與 1 之間。

```
Algorithm Similarity Evaluation
Input: Fine segment result U and ground truth V
Output: Similarity between U and V
1.   if m ≧ n then {
2.        for i = 0 to m do S[i, 0] = 0;
3.        for j = 1 to n do S[0, j] = 0;
4.        for i = 1 to m do
5.             for j = 1 to n do
6.                  S[i, j]=max(S[i-1, j-1] + d(ui, vj), S[i-1, j] ); }
7.   else {
8.        for i = 1 to m do S[i, 0] = ∞;
9.        for j = 0 to n do S[0, j] = 0;
10.       for i = 1 to m do
11.            for j = 1 to n do
12.                 S[i, j]=max( S[i-1, j-1] + d(ui, vj), S[i, j-1] ); }
13.  return S[m, n]
```

圖 5.4：相似度評估的演算法。

U 與 V 比較相似度時，會有三種可能的情況。第一種，U 與 V 的段落數目相同；

第二種，U 的段落數大於 V 的段落數；第三種，U 的段落數小於 V 的段落數。一般我們在 Normalize 時，會直接除以 $\sqrt{m}$ 與 $\sqrt{n}$，但是在第三種情形時，U 的段落越少，Normalize 的結果確會比較好。為了避免這種情形的發生，我們將 Normalize 的方法改定義為 $\frac{\text{Similarity}}{\sqrt{m}\sqrt{\max(m,n)}}$。我們剛剛求得的 U 與 V 相似度 104/30 必須再除以 $\sqrt{5}\sqrt{\max(5,4)}$，得到最後的結果為 52/75。

		(1,9)	(10,13)	(14,29)	(30,33)	(34,58)
	0	0	0	0	0	0
(1,4)	0	20/30	20/30	20/30	20/30	20/30
(5,29)	0	10/30	32/30	44/30	44/30	44/30
(30,33)	0	0	10/30	32/30	74/30	74/30
(34,58)	0	0	0	10/30	32/30	104/30

圖 5.5：計算 S(U, V)所需的矩陣 S 範例

5.3 實驗結果

本次實驗執行的硬體設備為 Intel Pentium 4 2.4G 的 CPU，以及 2G 的實體記憶體空間。以下我們將資料分別以 Original duration 與 IOI 兩種不同的方法表達旋律，找出個別的粗略段落、動機以及最後的精細段落。我們利用 5.2 所提出的評估方法，評估最後的實驗結果。表 5.2 為 Original duration 表示節奏的實驗結果。其中，效果最好的為莫札特的土耳其進行曲，與 Ground truth 的相似度為 0.84，透過我們方法找出來的段落有 17 段，正確的結果有 27 段。效果最差的為布拉姆斯的匈牙利舞曲，與 Ground truth 的相似度為 0.35，透過我們方法找出來的段落有 7 段，正確的結果有 20 段。表 5.3 為 IOI 表示節奏的實驗結果。其中，效果最好的為芭達潔芙絲卡的少女的祈禱，與 Ground truth 的相似度為 0.82，透過我們方法找出來的有 5 段，Ground truth 有 6 段。效果最差的為布拉姆斯的匈牙利舞曲，與 Ground truth 的相似度為 0.35，透過我們方法找出來有 3 段，正確結果有 20 段。

表 5.2：以 Original duration 來表示音樂節奏的實驗結果。

作曲者	曲名	**Result**
海頓	小夜曲	**0.72**
莫札特	土耳其進行曲	**0.84**
柴可夫斯基	胡桃鉗序曲	**0.51**
芭達潔芙絲卡	少女的祈禱	**0.64**
布拉姆斯	圓舞曲	**0.81**
莫札特	小步舞曲	**0.64**
貝多芬	給愛麗斯	**0.83**
巴哈	聖母頌	**0.58**
布拉姆斯	匈牙利舞曲	**0.35**
韋瓦第	春之頌	**0.59**

表 5.3：以 IOI 來表示音樂節奏的實驗結果。

作曲者	曲名	**Result**
海頓	小夜曲	**0.63**
莫札特	土耳其進行曲	**0.75**
柴可夫斯基	胡桃鉗序曲	**0.51**
芭達潔芙絲卡	少女的祈禱	**0.82**
布拉姆斯	圓舞曲	**0.75**
莫札特	小步舞曲	**0.64**
貝多芬	給愛麗斯	**0.80**
巴哈	聖母頌	**0.43**
布拉姆斯	匈牙利舞曲	**0.35**
韋瓦第	春之頌	**0.51**

第六章

結論與未來研究

6.1 結論

我們利用音樂的動機輔助對音樂作主題式的分段，進一步提供使用者以主題段落瀏覽音樂。由於音樂的主題必須透過音樂內容做分析，所以我們探勘出音樂的動機，改善在主題分段時的效果。

我們利用構成音樂主題最重要的因素動機來輔助主題的分段。我們提出一個以主題分段的方法，分為以下四個流程。首先，針對一首音樂，我們以 All-mono 的技術擷取出音樂的主旋律，以供分析。我們採用兩種不同的方法來表達主旋律的節奏，分別是保留原始音長的 Original Duration 與以開始發音時間為間隔的 IOI 表示法。第二，我們修改探勘 Non-trivial 重複樣式的探勘方法，對主旋律做粗略的分段，粗略分段的結果可以用來探勘動機，以及在最後精確分段的參考。第三，我們利用粗略分段的結果，以 Motivic treatment 的方法探勘動機。我們共考慮 Repetition，Transpose，Sequence，Contrary motion，Retrograde 與 Augmentation or Diminution 等六種動機變化。分別利用這六種不同的特性，修改傳統找重複樣式的技術，用以探勘符合動機變化的重複樣式。第四，找出動機後，我們用粗略分段的結果與動機出現的位置，判斷是否要對一個粗略分段做 Split 或將兩個粗略分段做 Merge，產生出 Theme segments。找出所有的 Theme segments 後，音樂中剩下的 Non-theme segments 會分別檢查長度是否超過一個定值。如果長度超過該定值，也將符合條件的 Non-theme segment 保留在最後的結果內。最後的結果，即可以用來做主題的瀏覽。

我們提出一個新的評估分段效果的方法，改進以前只考慮 Precision 或 Recall 所可能造成的缺點。傳統 Precision 或 Recall 的方法不能判斷相似的程度，用在音樂主題分段結果的評估並不適合。我們提出來的評估方法，找出正確段落的答案與透過我們方法的段落結果之間，找一個最佳的對應，並且計算每個對應段落之間的相似度，得到最後的結果。

最後，我們針對十首音樂，以 Original duration 以及 IOI 兩種不同表示節奏的方法分別做實驗，並且以我們提出來的評估方法做結果的評估。

6.2 未來研究

除了以 Motivic treatment 的方法來探勘動機外，我們可以用其他的方法來探勘動機。LBDM 的技術可以將音樂分解成許多的音型(Figure)，動機可以由一個以上的音型所構成。所以我們可以先利用 LBDM 的技術將粗略分段的結果分解成很多個音型。然後利用 Clustering 的技術將這些音型做分群，同一個群組內的音型即可能與同一個動機相關。我們在分群的時候，計算音型之間的距離，也可以利用動機變化的六種情形。

除了修改傳統探勘重複續列的方法以符合動機特性外，我們可以進一步訂立出符合動機特性而且與一般重複序列特性不同的的動機樣式(Motive pattern)，以及探勘的方法，以改進效率或效果。

直接利用探勘出來的動機，我們也可以自動產生出音樂的摘要(Summary 或 Thumbnailing)。因為動機代表音樂的主題，摘要必須呈現音樂中最主要的部份。所以透過產生出來的動機，我們可以結合成音樂的摘要。

此外，動機也可以作為以音樂內容查詢(Content based music retrieval)時，比對的資料。相對於對整首音樂作查詢比對，音樂的動機可以改進比對時的效率。

參考文獻

[1] M. A. Bartsch and G. H. Wakefield, "To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing," *Proc. of Workshop on Applications of Signal Processing to Audio and Acoustics*, *WASPAA'01*, 2001.

[2] R. Bod, "A Memory-Based Model for Music Analysis: Challenging the Gestalt Principles," *Journal of New Music Research*, *Vol. 31, pp. 27-37*, 2001.

[3] E. Cambouropoulos, "A Formal Theory for the Discovery of Local Boundaries in a Melodic Surface," *Proc. of the III Journees d' Informatique Musicale*, 1996.

[4] E. Cambouropoulos, "The Local Boundary Detection Model (LBDM) and its Application in the Study of Expressive Timing," *Proc. of the International Computer Music Conference, ICMC'01*, 2001.

[5] W. Chai and B. Vercoe, "Folk Music Classification Using Hidden Markov Models," *Proc. of the International Conference on Artificial Intelligence*, *ICAI'01*, 2001.

[6] W. Chai and B. Vercoe, "Music Thumbnailing via Structural Analysis," *Proc. of ACM International Conference on Multimedia*, *MM'03*, 2003.

[7] W. Chai and B. Vercoe, "Structural Analysis of Musical Signals for Indexing and Thumbnailing," *Proc. of ACM/IEEE Joint Conference on Digital Libraries, JCDL'03*, 2003.

[8] Y. J. Chen, "A Fast Repeating Pattern Finding Algorithm for Music Data: A Human Perceptive Approach," Master Thesis, Department of Electrical Engineering, National Cheng Kung University, Taiwan, 2004.

[9] D. Conklin and C. Anagnostopoulou, "Representation and Discovery of Multiple Viewpoint Patterns," *Proc. of the International Computer Music Conference*, *ICMC'01*, 2001.

[10] R. B. Dannenberg, "A Brief Survey of Music Representation Issues, Techniques and Systems," *Computer Music Journal, Vol. 17, No. 3, pp. 20-30*, 1993.

[11] R. B. Dannenberg, B. Thom, and D.Watson, "A Machine Learning Approach to Musical Style Recognition," *Proc. of International Computer Music Conference*, *ICMC'97*, 1997.

[12] C. David, Virtual Music : Computer Synthesis of Musical Style, MIT Press, 2001.

[13] S. Dixon, E. Pampalk, and G. Widmer, "Classification of Dance Music by Periodicity Patterns," *Proc. of International Symposium on Music Information Retrieval*, *ISMIR'03*, 2003.

[14] M. Grachten, J. L. Arcos and R. L. Mántaras, "Melodic Similarity: Looking for a Good Abstraction Level," *Proc. of International Symposium on Music Information Retrieval*, *ISMIR'04*, 2004.

[15] K. Hirata and S. Matsuda, "Interactive Music Summarization based on GTTM," *Proc. of International Symposium on Music Information Retrieval*, *ISMIR'02*, 2002.

[16] J. L. Hsu, "Content-based Music Information Retrieval and Analysis," Ph.D. Thesis, Department of Computer Science, National Tsing Hua University, Taiwan, 2001.

[17] J. L. Hsu, C. C. Liu, and A. L. P. Chen, "Discovering Nontrivial Repeating Patterns in Music Data," *IEEE Transactions on Multimedia*, *Vol. 3, No. 3, pp. 311-325*, 2001.

[18] J. L. Hsu, C. C. Liu, and A. L. P. Chen, "Efficient Repeating Pattern Finding in Music Database," *Proc. of Conference on Information and Knowledge Management*, *CIKM'98*, 1998.

[19] C. S. Iliopoulos, K. Lemström, M. Nuyad, and Y. J. Pinzón, "Evolution of Musical Motifs in Polyphonic Passages," *Proc. of Symposium on AI and Creativity in Arts and Science*, *AISB'02*, 2002.

[20] C. S. Iliopoulos, T. Lecroq, L. Mouchard, and Y. Z. Pinzon, "Computing Approximate Repetitions in Musical Sequence," *International Journal of Computer Mathematics*, V*ol.3, No. 1&2*, 2002.

[21] J. S. R. Jang, H. R. Lee, C. H. Yeh, "A Query-by-Tapping System for Music Retrieval," *Proc. of IEEE Pacific-Rim Conference on Multimedia*, 2001.

[22] M. H. Jian, C. H. Lin, and A. L. P. Chen, "Perceptual Analysis for Music Segmentation," *Proc. SPIE Storage and Retrieval Methods and Applications for Multimedia*, 2004.

[23] I. Koprinska and S. Carrato, “Temporal Video Segmentation: A Survey,” *Signal Processing Image Communication*, *Vol. 16, No. 5, pp. 477-500*, 2001.

[24] F. F. Kuo, “Melody Style Mining Using Chord Features,” Master Thesis, Department of Computer Science, National Chengchi University, Taiwan, 2003.

[25] O. Lartillot, “Discovering Musical Patterns through Perceptive Heuristics,” *Proc. of International Symposium on Music Information Retrieval*, *ISMIR’03*, 2003.

[26] O. Lartillot, S. Dubnov, G. Assayag, and G. Bejerano, “Automatic Modeling of Music Style,” *Proc. Of International Computer Music Conference*, *ICMC*, 2002.

[27] F. Lerdahl and R. Jackendoff, A Generative Theory of Tonal Music, MIT Press, 1983.

[28] C. C. Liu, J. L. Hsu, and A. L. P. Chen, “Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases,” *Proc. of International Conference on Data Engineering*, *ICDE’99*, 1999.

[29] B. Logan and S. Chu, “Music Summarization Using Key Phrases,” *Proc. of International Conference on Speech, Acoustics, and Signal Processing*, *ICASSP’00*, 2000.

[30] L. Lu, H. You, and H.J. Zhang, “A New Approach to Query by Humming In Music Retrieval,” *Proc. of IEEE International Conference on Multimedia and Expo*, *ICME’01*, 2001.

[31] K. D. Martin, E. D. Scheirer, and B. L. Vercoe, “Music content analysis through Models of Audition,” *Proc. of ACM Multimedia Workshop on Content-Based Processing of Music*, 1998.

[32] C. Meek and W. P. Birmingham, “Automatic Thematic Extractor,” *Proc. of International Symposium on Music Information Retrieval*, *ISMIR’01*, 2001.

[33] C. Meek and W. P. Birmingham, “Automatic Thematic Extractor,” *Journal of Intelligent Information Systems, Vol. 21, No. 1, pp. 9-33*, 2003.

[34] D. Meredith, K. Lemström, and G. A. Wiggins, “Algorithm for Discover Repeating Patterns in Multidimensional Representations of Polyphonic Music,” *Journal of New Music Research*, *Vol.31, No.4, pp. 321-345*, 2002.

[35] J. Pickens, “Feature Selection for Polyphonic Music Retrieval,” *Proc. of the ACM*

International Conference on Research and Development in Information Retrieval, *SIGIR'01*, 2001.

[36] D. Rizo, J. M. Iñesta, F. Moreno-Seco, "Tree-structured Representation of Musical Information," *Iberian Conference on Pattern Recognition and Image Analysis*, 2003.

[37] M. K. Shan, F. F. Kuo, and M. F. Chen, "Music Style Mining and Classification by Melody," *Proc. of IEEE International Conference on Multimedia and Expo*, *ICME'02*, 2002.

[38] M. K. Shan and F. F. Kuo, "Music Style Mining and Classification by Melody," *IEICE Transactions on Information and System, Vol. E86-D, No. 4*, 2003.

[39] H. H. Shih, S. S. Narayanan, and C. C. J. Kuo, "A Dictionary Approach to Repeating Pattern Finding in Music," *Proc. of IEEE International Conference on Multimedia and Expo*, *ICME'01*, 2001.

[40] C. Spevak, B. Thom, and Karin Höthker, "Evaluating Melodic Segmentation," *Proc. of International Conference on Music and Artificial Intelligence*, *ICMAI'02*, 2002.

[41] L. Stein, Structure & Style：The Study and Analysis of Musical Forms, Summy-Birchard Music, 1979.

[42] D. Temperley, The Cognition of Basic Musical Structires, MIT press, 2001.

[43] B. Thom, "BoB: An Improvisational Music Companion," Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, USA, 2001.

[44] B. Thom, C. Spevak, and K. Höthker, "Melodic Segmentation: Evaluating the Performance of Algorithms and Musical Experts," *Proc. of the International Computer Music Conference, ICMC'02,* 2002.

[45] Y. H. Tseng, "Content-Based Retrieval for Music Collections," *Proc. of ACM Conference on Research and Development in Information Retrieval, SIGIR'99,* 1999.

[46] A. L. Uitdenbogerd and J. Zobel, "Manipulation of Music For Melody Matching," *Proc. of ACM International Conference on Multimedia*, *MM'98*, 1998.

[47] A. L. Uitdenbogerd and J. Zobel, "Melodic Matching Techniques for Large Music Databases," *Proc. of ACM International Conference on Multimedia*, *MM'99*, 1999.

[48] T. Weyde, “Integrating Segmentation and Similarity in Melodic Analysis,” *Proc. of the International Conference on Music Perception and Cognition*, *ICMPC’02*, 2002.

[49] C. Xu, Y. Zhu, and Q. Tian, “Automatic Music Summarization Based on Temporal, Spectral and Cepstral Features,” *Proc. of IEEE International Conference on Multimedia and Expo*, *ICME’02*, 2002.

[50] M. Zinn and R. Hogenson, Basic of Music: Opus 1, Schirmer books, 1994.

計畫成果自評

■ 就研究內容與原計畫相符程度、達成預期目標情況

本研究計畫已經完成音樂分段技術，以提供音樂瀏覽之技術研究。本計畫利用音樂的動機輔助對音樂作主題式的分段，進一步提供使用者以主題段落瀏覽音樂。由於音樂的主題必須透過音樂內容做分析，所以我們探勘出音樂的動機，改善音樂主題分段的效果。此外，我們提出一個新的評估分段效果的方法，改進以前只考慮 Precision 或 Recall 所可能造成的缺點。本計畫因為實驗所需之音樂，在音樂轉譯時品質不佳，因此申請計畫延期。在延期三個月中，我們已經完成音樂的實驗。實驗結果顯示我們所研究的音樂分段效果準確率最高可以到 84%，平均 65%。

■ 研究成果之學術或應用價值

本計畫的研究成果，在學術價值方面，我們提出新的音樂動機探勘的演算法。我們也提出 split and merge 的方法來找出音樂的主題片段。本研究成果也將有助於音樂結構分析(music structure analysis)與音樂摘要產生(music summarization)的相關研究。
在應用價值方面，本計畫所開發的技術可利用於文化、娛樂與音樂教育產業、智慧型音樂檢索系統中音樂瀏覽、主題試聽的功能。也可應用在音樂學習軟體，以幫助使用者對音樂結構及音樂動機的了解。

■ 是否適合在學術期刊發表或申請專利

本計畫的研究成果除了兩篇碩士論文之外，部分研究成果已經投稿至國際學術會議 2006 年的 IEEE Syetems, Man and Cybernetics (IEEE SMC)。此外，我們也預計將部分研究成果整理後，投稿至 2007 年的 ACM/IEEE Joint Conference on Digital Libraries (JCDL)。我們也預計將這些研究成果整理投稿至學術期刊。

■ 主要發現或其他有關價值

本計畫利用所研究的音樂分段技術，分析音樂分段的效果。根據實驗結果顯示，莫札特、貝多芬等音樂家的作品，其分段效果優於其他音樂家的作品。此實驗結果可能的原因，或與貝多芬、莫札特後期的作曲方式有關。他們皆以音樂動機為基礎來之發展其作品。而本研究的音樂分段技術，先利用動機探勘技術，分析音樂動機後，在利用動機找出音樂主題。由音樂主題將音樂分段。

參與本計畫的碩士班研究生，目前有一位同學就讀於國立交通大學資訊工程研究所博士班，繼續進行電腦音樂的相關研究。一位目前於奇美通訊工作，負責手機音樂的相關軟體開發。另一位同學於華寶通訊工作，負責手機軟體的開發。一位同學目前擔任國立台灣大學原住民數位典藏計畫專任研究助理。

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：95 年 5 月 31 日

國科會補助計畫	計畫名稱：數位音樂典藏之資料探勘與智慧型檢索技術 計畫主持人：沈錳坤 計畫編號：NSC 94-2422-H-004-003-　學門領域：資訊工程
技術/創作名稱	以音樂動機為基礎之音樂分段技術
發明人/創作人	沈錳坤
技術說明	中文： 我們主要提出音樂的主題分段技術。 音樂主題分段技術總共分為四個主要的步驟。 第一：我們從原始的音樂資料擷取出主旋律的部分。 第二：將主旋律做粗略分段。我們利用探勘 Non-trivial 重複樣式的技術，來找出粗略段落。 第三：從粗略段落中探勘動機。我們利用 Stein 所提出來的動機變化規則，修改傳統探勘重複序列的方法，做動機的探勘。 最後：我們利用探勘出來的動機對主旋律作精細分段。
	英文： We proposed the theme segmentation for digital music. There are four steps in the theme segmentation technique. Firstly, we extract main melody from original music. In the second step, rough segments are generated from main melody by mining non-trivial repeating patterns. Then, motives are detected from rough segments. We modify the mining algorithm for discovering frequent patterns by applying motivic treatment rules proposed by Stein. Finally, we segment main melody based on the generated motives.
可利用之產業及可開發之產品	1. 可利用於文化、娛樂與音樂教育產業 2. 智慧型音樂檢索系統，在網站中提供對音樂的瀏覽、音樂主題的試聽 3. 音樂學習軟體，幫助使用者對音樂結構及音樂動機的了解
技術特點	1. 運用資料探勘技術，對音樂內容分析。 2. 提出以音樂動機為基礎的音樂分段技術
推廣及運用的價值	1. 大量音樂典藏或資料庫中，音樂分段技術將可提供使用者瀏覽音樂的功能。 2. 音樂分段技術也有助於與其他媒體的音樂配樂。