A SONG INQUIRY SYSTEM USING MULTI-COMPONENT DYNAMIC TIME WARPING

尤靖允、鄭群瀚

專題摘要

本報告詳述了如何使用 deep discrete Fourier transform(DDFT)演算法取出一段音樂的音高組成，並以這樣多重音高的資料形式，設計出相應的 multi-component dynamic time warping(MDTW)演算法來實作歌曲搜尋的系統。在一個小型的資料庫上實驗之後，我們發現不管是用真實樂手演奏或用軟體樂器重新產生的樂曲，其 MRR 數值都有不錯的成績。

1.動機及目標

在做實驗七「哼唱式歌曲查詢系統」時，我們發現助教給的 wav2note 程式只能輸出單一音高，如果輸入的訊號同時有不同音高的話就會抓不出來。實驗時影響不大，是因為系統設計成用人哼唱的聲音當輸入源，不會有超過一種音高的情形。於是我們想，如果加入處理多重音高的能力，是不是就能做出一個用途更廣泛的系統呢？這就是我們最初的動機。

而這樣的系統能用在甚麼地方呢？在一個樂團表演的現場，常常會有聽到一首歌覺得很好聽，想查詢時，卻因為不知道曲名而無從下手的狀況。這時我們的系統就可以分析當下的聲音，比對出最適合的歌曲。由於樂團現場的演奏一定會跟原版有一些出入，實驗七用到的 DTW 就可以處理這些時間上的不同。我們希望做出來的系統，能在相當程度上有這樣的功能。

2.專題內容

2.1 系統架構

我們的系統分成兩個部分：(1)音樂的採譜。(2)DTW 的改寫。採譜的部分使用了尤姓組員和中研院蘇黎助理研究員實習時做的題目 deep discrete Fourier transform(DDFT)來處理，DTW 則是從實驗七的程式去做修改。

2.2 DDFT

DDFT 的概念大致上，是將傅立葉轉換與類神經網路做類比，將每一次的轉換都視成是一層網路 [1]。可表示成：

$$\boldsymbol{Z}^{(0)} = \sigma^{(0)}(|\boldsymbol{X}|)\ ,\ \boldsymbol{Z}^{(l)} = \sigma^{(l)}\left(\boldsymbol{W}^{(l)}\boldsymbol{F}\boldsymbol{Z}^{(l-1)}\right)\ ,\ l \geq 1$$

其中 **X** 代表時頻譜，**F** 代表傅立葉轉換並只取實部，**W** 則是一個高通濾波器，σ 是一個非線性轉換。**Z** 可想成是一層類神經網路，其數值是前一層網路經過非線性轉換、濾波、傅立葉轉換的結果。非線性轉換會將數值較小的部分放大，而高通濾波器則是將過低的頻率和過低的週期濾掉。結果就是隨著層數加深，音色資訊越來越不明顯，凸顯出音高資訊。然後取最後兩層的資料，找出泛音序列和次泛音序列重疊的部分就是基音 [2]。

2.3 MDTW

MDTW 的運算和一般的 DTW 相同，但是兩個序列點與點之間距離的運算要做變更，因為原本的運算只考慮到單一數值而非多重數值的情形。一開始想到的做法是先將兩個陣列的長度調整成一樣後，嘗試所有不同數值對應的情形，最後取距離最小的組合當作結果。舉一個例子說明。有兩個不同長度的陣列，X 和 Y 分別取自兩個序列的其中一個音框，數值為[36, 43]和[24, 36, 43]。如果是 MIDI 的話，我們可以看出來 X 是由 C2 和 G2 組成的五度音程，Y 則是多了一個 C1。如果這兩個陣列都代表同一首曲子的同個時間狀態時，有可能是 X 的序列的原始演奏者少彈一個音，或是 Y 的演奏者多彈一個音，又或者是程式的誤判，畢竟平行八度在音高辨識是一個較難區別的情形。首先我們將 X 做 zero-padding，使它的長度和 Y 相同，接著嘗試 X 和 Y 所有可能的對應組合，相減取絕對值加總，取最小的組合；如果相減的其中一個數值為 0，表示這個值是多出來的，我們可以給他一個處罰值 ε。可表示成：

$$\mathrm{d(X,Y)} = \min_{X' \epsilon X_{n!}} \left(\sum_{i=1}^{n} \begin{cases} \varepsilon, & if\ x'_i\ or\ y_i = 0. \\ |x'_i - y_i|, & o.w. \end{cases} \right),$$

其中$X_{n!}$表示 X 所有元素的排列組合，n 則為 zero-padding 後兩個陣列的長度。以上例子找出來距離組合如 Figure 1，其距離 d(X, Y)=0+0+ε=ε。

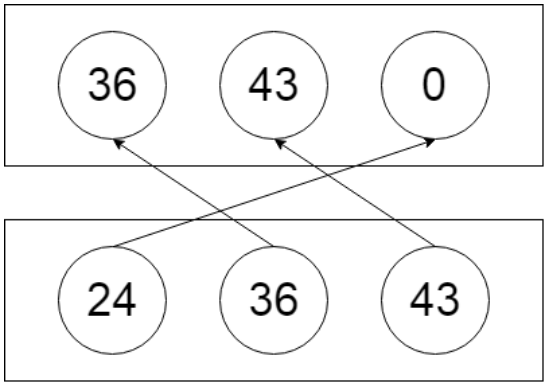

Figure 1. All permutation method.

但如果 n 很大時，嘗試 n!種組合的運算量就非常可觀，所以我們改成另一種較簡單的算法：將兩個陣列排序，取長度最短的陣列長度，相減取絕對值加總。例子的距離就會變成 d(X, Y)=|36-24|+|43-36|=19，組合如 Figure 2。不僅運算方便，我們也不用特別去定義處罰值 ε，因為當一個陣列有多出來的元素時，他的距離一定會增加。

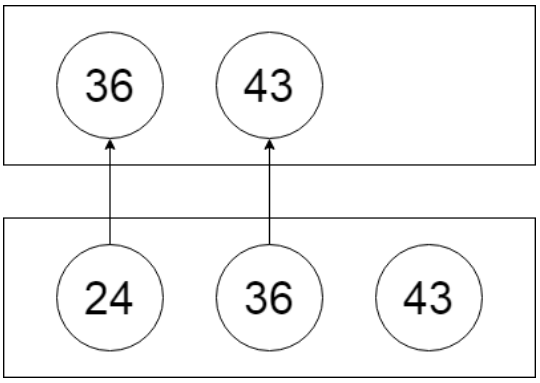

Figure 2. Sorting method.

2.4 資料庫

在測試資料的選擇上，我們找了 Bach10 [3]這個資料庫做測試，共有十首由真實樂手演奏的巴哈四重奏。而資料庫也有曲子的 MIDI 檔，我們可以用調整這些 MIDI，產生出不同演奏版本，並用軟體樂器重新輸出成音訊再去取音高。

2.5 實驗流程

首先必須有足夠多不同版本的曲子來當實驗資料。除了 Bach10 原本附的音高資

料外，又用 DDFT 抓了另一個版本，其輸出可當作同一首曲子的變體[1]。接著用 MIDI 增加三種版本：(1)原始版，bpm 設成 100。(2)人性化版，將音符的起始時間與長度加上一些隨機數值，使之較符合真實世界演奏的狀況，bpm 也是 100。(3)慢速版，將前一版的 bpm 設成 75。再從音源庫 kontakt factory library [4]找到對應的軟體音色(bassoon，violin，clarinet 和 saxphone)，將 MIDI 重新輸出成音訊。於是我們有了十首曲子的五種不同版本，共 50 個樣本的資料庫，美個樣本長度介於 20 秒至 60 秒之間。

接著將音訊樣本用 DDFT 轉成 M x N 的二維陣列(原本的音高資料除外)，N 為音框數量，M 則是一個大於 1 的變數(因為音高的數量並不固定)，量值為 MIDI 的 pitch。DDFT 的窗函數大小=7939，音框間格=10 毫秒，非線性轉換參數$[\gamma_0, \gamma_1, \gamma_2, \gamma_3] = [0.1, 0.9, 0.9, 0.5]$ (四層傅立葉轉換)。

最後從五種版本裡取兩種，一種當作對照樣本，另一種當作受測樣本，總共 20 種組合，依照以下公式計算 Mean Reciprocal Rank (MRR)，q_i為第 i 個受測資料，rank 則是 MDTW 的相似度排名。MDTW 的計算上我們試了 type-1 與 type-2 兩種不同的移動方式，global constraint 為 end-to-end，也就是路徑的開始與結束都要與受測樣本及對照樣本相同。

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank(q_i)}$$

2.6 使用軟體與環境

所有程式都在 Ubuntu 16.04 LTS 系統運行，DDFT 的語言為 python 2.7，MDTW 則是用 C++。由於鄭姓組員對於 Python 相對不熟悉，且 DDFT 是現成拿來用的程式，主要還是用 C++於組員之間的討論，著重較多部分在 MDTW 的改良。MIDI 的調整則在 Sonar X3 Studio [5]進行，並用 Kontakt [4]的音色輸出。

[1] 在未公開的論文實驗裡，DDFT 在相同的資料庫上其音高 f1-score 評分可達 86.5%。

3.實驗結果及分析

實驗結果為 Table 1 至 Table 3，可以看出來大致上 type-1 的表現比 type-2 好不少。這其實超出我們的預期，因為我們想說 type-1 的移動角度沒有 type-2 那麼大，如果有變化快速的地方 type-1 應該會跟不上。從結果上，只能說 type-1 的路徑選擇會比 type-2 好，至少在這個資料庫上是如此。另外可以看到如果對照資料的速度和受測資料相差太多，會影響 MDTW 表現。使用 MIDI 生成樣本之間的互評分數都是 100%，可能是因為用軟體音源的關係，和真實錄音的情況相比變數較少。

Table 1. MDTW(type-1)實驗結果。

Type-1		**對照樣本**				
		原始音高	DDFT	原始 MIDI	人性化 MIDI	慢速 MIDI
受測樣本	原始音高	x	95.00%	95.00%	95.00%	73.33%
	DDFT	95.00%	x	100.00%	100.00%	90.00%
	原始 MIDI	95.00%	100.00%	x	100.00%	100.00%
	人性化 MIDI	95.00%	100.00%	100.00%	x	100.00%
	慢速 MIDI	95.00%	100.00%	100.00%	100.00%	x

Table 2. MDTW(type-2)實驗結果。

Type-2		**對照樣本**				
		原始音高	DDFT	原始 MIDI	人性化 MIDI	慢速 MIDI
受測樣本	原始音高	x	90.00%	78.67%	78.33%	64.17%
	DDFT	95.00%	x	91.67%	84.50%	84.33%
	原始 MIDI	95.00%	78.75%	x	100.00%	100.00%
	人性化 MIDI	95.00%	82.36%	100.00%	x	100.00%
	慢速 MIDI	95.00%	83.10%	100.00%	100.00%	x

Table 3. 各對照樣本的 MRR 平均與總平均。

	原始音高	DDFT	原始 MIDI	人性化 MIDI	慢速 MIDI	總平均
Type-1	**95.00%**	**98.75%**	**98.75%**	**98.75%**	**90.83%**	**96.42%**
Type-2	**95.00%**	83.55%	92.58%	90.71%	87.13%	89.79%

4.結論

我們驗證了本系統在搜尋古典樂曲上的表現，且以 type-1 最好。雖然實驗結果還不錯，但我們認為實驗的樣本數量及變化量還不夠多。也許之後可以試著將 MIDI 的數值平移幾度，或是把一兩樣樂器抽離，試試看會對系統的表現有怎樣的影響，或者是換成跟原本完全不一樣的演奏樂器。畢竟真實情況下，可能會有用不同的樂器演奏，因為音域的關係移調，或是無法以完整樂器配置表演的情形。這些都是能考慮進來的因素。

參考文獻

[1] L. Su, "Between homomorphic signal processing and deep neural networks: constructing deep algorithms for polyphonic music transcription," in *Asia Pacific Signal and Information Processing Association*, 2017.

[2] L. Su, Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE Trans. Audio, Speech, Lang. Proc*, vol. 23, no. 10, pp. 1600-1612, 2015.

[3] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Trans. Audio, Speech, Lang. Proc*, vol. 18, no. 8, pp. 2121-2133, 10 2010.

[4] "Kontakt 5," Native Instruments, [Online]. Available: https://www.native-instruments.com/en/products/komplete/samplers/kontakt-5/. [Accessed 2017].

[5] "SONAR," Cakewalk, [Online]. Available: https://www.cakewalk.com/Products/SONAR. [Accessed 2014].