

國 立 中 央 大 學

資 訊 工 程 學 系
碩 士 論 文

小提琴演奏追蹤系統：應用音源分離結果實現即時音樂追蹤與伴奏

A Violin Performance Tracking System: Utilizing Music Source Separation Results for Real-Time Music Tracking and Accompaniment

研究 生：林妤潔

指 導 教 授：蘇木春 博 士

中 華 民 國 一 百 一 十 三 年 六 月

小提琴演奏追蹤系統：應用音源分離結果實現即時音樂追蹤與伴奏

摘要

本研究旨在開發一套結合音源分離模型的小提琴演奏追蹤系統，以實現小提琴與鋼琴的混合音源分離，並利用分離音源達成不同特徵下的音樂追蹤。

本研究設計了音源分離模組與音樂追蹤模組，在音源分離模組方面，我們自行蒐集並建立一套新的公開整合資料集，用於訓練 Band-Split RNN 模型，並改進了模型的頻帶切割方法。在模型的評估上，我們使用訊號失真比 (SDR) 來計算模型的分離效果，結果顯示模型在資料缺乏與資料充足的情況下皆優於現有的基線模型，並證明頻帶切割方法的有效性。在音樂追蹤模組方面，我們改進了線上動態時間規整演算法與貪心向後對齊方法，重現了即時音樂追蹤模組的設計，並改良部分元件。在實際的測試中，即時音樂追蹤系統展現了低延遲與精準追蹤的表現，並在不同特徵的表現上保持了穩定的追蹤效果。

關鍵字：音樂資訊檢索，音源分離，音樂追蹤，自動伴奏，深度學習，線上動態時間規劃

A Violin Performance Tracking System: Utilizing Music Source Separation Results for Real-Time Music Tracking and Accompaniment

Abstract

This study aims to develop a violin performance tracking system incorporating a source separation model to achieve mixed source separation of violin and piano, and to utilize the separated sources for music tracking under different features.

We designed a source separation module and a music tracking module. For the source separation module, we collected and established a new public integrated dataset to train the Band-Split RNN model, and improved the model's band-split method. In model evaluation, we used Signal-to-Distortion Ratio (SDR) to measure the separation performance. The results show that the model outperforms existing baseline models in both data-limit and data-rich cases, demonstrating the effectiveness of the band-split method.

For the music tracking module, we improved the Online Dynamic Time Warping algorithm (ODTW) and the Greedy Backward Alignment method(GBA), reimplementing the design of the real-time music tracking module and enhancing some blocks. In practical tests, the real-time music tracking system exhibited low latency and accurate tracking performance, maintaining stable tracking effects under different features.

Keywords: Music Information Retrieval, Music Source Separation, Music Tracking, Automatic Accompaniment, Deep Learning, Online Dynamic Time Warping

誌謝

誌謝...

目錄

	頁次
摘要	i
Abstract	ii
誌謝	iv
目錄	v
一、 緒論	1
1.1 研究動機	1
1.2 研究目的	3
1.3 論文架構	4
二、 背景知識以及文獻回顧	5
2.1 背景知識	5
2.1.1 小提琴與鋼琴的演奏特性	5
2.1.2 小提琴與鋼琴的音色分析	6
2.2 文獻回顧	9
2.2.1 音源分離相關研究	9
2.2.2 音樂追蹤相關研究	11
三、 研究方法	12
3.1 系統架構	12

3.2 音源分離模組	14
3.2.1 Band-Split RNN	14
3.2.2 估計不同樂器的頻帶切割點	15
3.3 音樂追蹤模組	17
3.3.1 動態時間規整演算法	17
3.3.2 線上動態時間規整演算法	19
3.3.3 貪心向後對齊方法	21
3.3.4 Data Manager Block	22
3.3.5 Music Detector Block	24
3.3.6 Rough Position Estimator Block	26
3.3.7 Decision Maker Block	28
四、 實驗設計與結果	31
4.1 音源分離評估	31
4.1.1 音源分離資料集	31
4.1.2 模型訓練細節	33
4.1.3 音源分離評估指標	33
4.1.4 音源分離結果比較	34
4.1.5 頻帶切割對於分離結果的影響	35
4.2 音樂追蹤評估	37
4.2.1 系統在不同速度下的追蹤結果	37
4.2.2 系統在使用音源分離音訊做為參考特徵的追蹤結果 ...	44
五、 總結	49
5.1 結論	49
5.2 未來展望	50
參考文獻	51

圖 目 錄

頁次

2.1 小提琴與鋼琴演奏 G4 音高三秒的特徵圖 上：小提琴， 中：鋼琴，下：混合	7
2.2 同一人演奏兩次 Beethoven Spring Sonata No.1 之特徵圖	8
2.3 不同人演奏 Bach BWV1006 之特徵圖	8
3.1 系統架構圖	12
3.2 BSRNN 架構圖，擷取自 [34]	14
3.3 兩種資料的波形圖 (上) 與經過 FFT 轉換後的 F_{band} 圖 (下)	16
3.4 兩個相似的時間序列：藍色： X 紅色： Y	17
3.5 DTW 累積距離矩陣回溯路徑示意圖	19
3.6 每個迭帶所計算到的距離成本矩陣格數位置比較	21
3.7 上：現場音訊的高 (左)/低 (右) 解析度特徵圖；下：參考 音訊的高 (左)/低 (右) 解析度特徵圖	24
3.8 RPE 距離矩陣	27
3.9 RPE 粗略估計位置	28
3.10 Decision Maker 計算的輸出點：淺藍：GBA 計算的最後輸 出位置 綠色：ODTW 計算的最後輸出位置	30
3.11 DTW 與音樂追蹤模組的輸出路徑比較圖	30
4.1 四種不同速度範圍的 MIDI 樂譜前五小節	38
4.2 系統在慢、一般速度的延遲時間	40

4.3 系統在快、加速速度的延遲時間	41
4.4 慢 (90-120bpm)	42
4.5 一般 (115-145bpm)	42
4.6 快 (135-175bpm)	43
4.7 加速 ($80 \rightarrow 160$ bpm)	43
4.8 110bpm 平均延遲：26.62 帖	45
4.9 120bpm 平均延遲：27.98 帖	45
4.10 130bpm 平均延遲：37.49 帖	46
4.11 140bpm 平均延遲：40.68 帖	46
4.12 自由速度平均延遲：28.43 帖	47
4.13 自由速度 (參考音訊為同一人所演奏) 平均延遲：22.54 帖 . .	48

表 目 錄

	頁次
3.1 STFT 參數設定	23
3.2 高解析度特徵參數設定	23
3.3 低解析度特徵參數設定	23
4.1 整合的訓練資料集格式	32
4.2 每種樂器的訓練集與驗證集大小	32
4.3 每次迭帶的訓練與驗證資料筆數	33
4.4 Data-limit 結果數值比較	34
4.5 Data-rich 結果數值比較	35
4.6 Data-limit 不同的 Band-Split Bandwidth 結果比較	36

一、緒論

1.1 研究動機

根據大學術科考試委員會 107 年至 112 年的音樂術科考試人數資料統計 [1]，樂器主修報考最多的項目分別為弦樂與鋼琴，其中弦樂主修又以小提琴佔比最高。由此可知，小提琴是許多人學習和演奏的樂器。在眾多涉及小提琴的樂曲中，除了無伴奏小提琴曲（例如巴赫無伴奏小提琴奏鳴曲等）之外，幾乎都需要與其他樂器合奏，而鋼琴則是最為常見的合奏樂器。例如小提琴奏鳴曲就是由小提琴與鋼琴共同演奏的曲子，此外也有許多曲目從原始樂器編制改編為鋼琴與小提琴的合奏版本。

通常演奏合奏曲目，必須自行尋找或是聘請其他演奏者共同演奏，但聘請其他演奏者的費用並不便宜，通常一節伴奏（約 50 分鐘到 1 小時）會根據專業度的不同來收費，平均收費約為台幣 600 元至 1800 元不等 [2]。因此若找不到其他演奏者，通常只能選擇演奏自己的部分或是在網路上尋找伴奏音訊。雖然有些演奏者會將音訊上傳至公開平台提供大家使用，但這些音訊通常已經是混合音訊，無法根據個人習慣或練習的速度演奏，也會被音訊中演奏同一部份的聲音干擾。

因此若能將網路上公開的混合音訊分離出伴奏音訊，並使伴奏音訊跟隨自己演奏的速度播放，便可以在無法找到其他演奏者時，自主練習合奏，同時也省下了人事成本。

近年來，隨著音樂數位化與深度學習的進步，音樂資訊檢索 (MIR) 這門領域的發展越來越受到許多人的關注，這門領域包含音樂來源分

離 [3]–[5]、自動伴奏 [6]–[8]、樂譜追蹤 [9], [10]、音樂生成 [11], [12]、樂器辨識 [13]–[15] 等子領域，這些研究領域也被廣泛的應用於商業化的產品上，例如音樂推薦系統 [16]、音樂創作工具 [17]–[19]、音樂教育應用軟體 [20], [21] 等。

國際音樂資訊檢索協會 (ISMIR) [22] 自 2000 年開始每年舉行 MIR 研討會，促進相關領域的交流。其中音樂來源分離與自動伴奏更是近幾年許多人關注的領域，音樂來源分離的主要目標為分離混和音訊中的各個音源 (樂器)，分離出來的音源可應用於音樂的重新混音，如卡拉OK、DJ 混音等，或作為其他問題的前處理工具 [23]。Sony 與 ISMIR 在 2021 年舉辦了音樂解混 (MDX) 競賽 [24]，大力推動了這項技術的發展，近期也延續了先前的成果舉辦更大的聲音分離 (SDX) 競賽 [25]。自動伴奏的主要目標為根據特定的旋律生成伴奏 [26], [27]，或是追蹤特定的旋律跟隨樂譜 [28]。這項技術已經在音樂教育與音樂創作 [29] 等商業產品使用。

在上述的兩個領域中，音樂來源分離的研究重點主要集中在流行樂音源的分離，自動伴奏使用的音訊資料大部分也是來自可通過 MIDI 協定傳輸的樂器。然而古典樂器因為音色的複雜性與資料的稀缺性，在音源分離與自動伴奏領域的研究相對較少，目前也尚未有將音源分離的結果結合於自動伴奏的研究，因此本研究旨在開發一套專注於追蹤小提琴演奏的即時音樂追蹤系統，此系統應用音源分離技術將混合音源分離為參考音訊與伴奏音訊，並作為即時音樂追蹤系統的參考音訊使用，讓個人練習時有方便的伴奏系統可以使用，也提供音樂創作者更多不同的創作方式。

1.2 研究目的

本研究的目的是開發一套專注於追蹤小提琴演奏的即時音樂追蹤系統，此系統分兩部分研究，第一部分為音源分離技術的研究，此研究專注於探討如何分離小提琴與鋼琴的混合音訊，作為後續音樂追蹤的參考音訊使用。研究詳細內容包含探討如何根據不同樂器的特性調整資料前處理的方式，以提升音源分離模型的效果、應用深度學習網路訓練音源分離模型並與過往表現突出的模型比較效果。第二部分為音樂追蹤技術的研究，此研究專注於探討如何使用來源不同的參考音訊來追蹤現場小提琴演奏的樂曲位置，並輸出對應的伴奏達到即時合奏的效果。研究詳細內容包含探討系統行程與線程的設計，平均分配系統計算資源來達到即時的效果、設計參考音訊與現場串流音訊的特徵提取方式、設計音樂偵測模組判斷現場演奏是否開始、改良粗略估計位置模組、線上動態時間規整演算法與貪心向後對齊方法提升追蹤位置的準確率、設計決策模組決定最後輸出位置並輸出對應的伴奏音訊。

1.3 論文架構

本論文分為五個章節，其架構如下：

第一章、緒論，敘述本論文之研究目的、動機以及架構。

第二章、背景知識以及文獻回顧，介紹本研究所需的背景知識，包含小提琴與鋼琴的基本演奏方式與特性分析，以及小提琴與鋼琴的音色比較，並探討目前在音源分離領域與音樂追蹤領域的研究現況。

第三章、研究方法，說明本研究細節，如整體的系統架構、音源分離模組設計、音樂追蹤模組設計與改良細節。

第四章、實驗設計與結果，說明實驗使用的資料集、實驗設計內容以及評估方法，並對於實驗結果進行探討。

第五章、總結，對於整體研究結果進行總結，提出尚可改進的部分並討論本研究的未來展望。

二、背景知識以及文獻回顧

2.1 背景知識

本研究專注於開發一套適用於小提琴與鋼琴的系統，因此本節將介紹小提琴與鋼琴的基本特性與特色。

2.1.1 小提琴與鋼琴的演奏特性

小提琴是一種弓弦樂器，演奏方式透過左手與右手的協調配合來實現。左手透過指腹按壓琴弦來調整音高，常見的左手指法技巧有滑音、抖音等等。滑音為通過指腹沿琴弦滑動到不同位置，產生音符連續變化的音色，抖音則是通過手腕的擺動，使按壓點的觸碰面積改變產生微小的音高波動，使音符聽起來有抖動的音色；右手握弓並將弓毛貼住琴弦，透過手臂與手腕的上下來回動作使弓與琴弦摩擦並發出聲音，常見的弓法技巧有連弓、跳弓等等。連弓技巧是在一弓之內演奏多個音符，產生連貫流暢的音色。跳弓則是使弓在琴弦上跳動，產生彈性與清晰的音色。

鋼琴是帶有琴弦的鍵盤樂器，演奏方式透過雙手與雙腳的協調配合來實現。雙手透過指腹敲擊琴鍵，使琴槌敲擊琴弦發出聲音，其中可以透過不同的敲擊力度與速度產生不同的音量與音色。雙腳透過踩踏鋼琴下的踏板，改變聲音的音色。踏板大致可分為延音踏板與弱音踏板，延音踏板可使音符響起的時間更長，而弱音踏板則是使彈奏的音符更為柔和。

由於樂器本身結構的不同，使兩者演奏的音色也大不相同，小提琴在演奏時，左手可以隨意控制音高，右手可以控制音量，且隨著演奏者的技巧能力，能展現非常多種且細膩的音色變化；鋼琴則是已經有固定的琴鍵控制音高，在音色方面可以使用不同力道敲擊琴鍵並配合踏板使用來表現，但與小提琴相比，音色變化的多樣性較低，但由於鋼琴為雙手彈奏的樂器，在多聲部或複雜和聲的表現上比起小提琴更能勝任這類的作品。

2.1.2 小提琴與鋼琴的音色分析

當兩種樂器以同樣的音量演奏同一段旋律，我們可以輕易地分辨哪一段旋律來自哪種樂器，甚至是當不同人以同樣的樂器、音量、旋律演奏，我們還是可以分辨出兩段旋律的差異。以物理學來看，聲音(聲波)是由頻率與振幅組成，頻率決定了音高，振幅決定了聲音的大小，而我們所聽到的聲音大部分是由許多不同頻率疊加在一起的，因此頻率的組成又可以拆成基頻與泛音，基頻決定了音高，泛音則是決定這個聲音的音色，也就是人類可以區分不同聲音的原因。

舉個例子，圖 2.1為小提琴與鋼琴演奏同一個音高 G4($\sim 392\text{Hz}$) 三秒鐘的特徵圖，上中下分別為鋼琴、小提琴與鋼琴小提琴合併後的特徵圖，在上圖與中間圖最亮的地方皆為頻率軸(y 軸) 上約 392Hz 的地方，也就是代表 G4 音高的基頻，而泛音通常會出現在基頻的整數倍，因此在頻率軸 $392 \times 2, 392 \times 3\dots\text{Hz}$ 的地方也有特徵顯示。可以看到兩種樂器在泛音頻率的成份並不相同，因此造就了不同的音色。

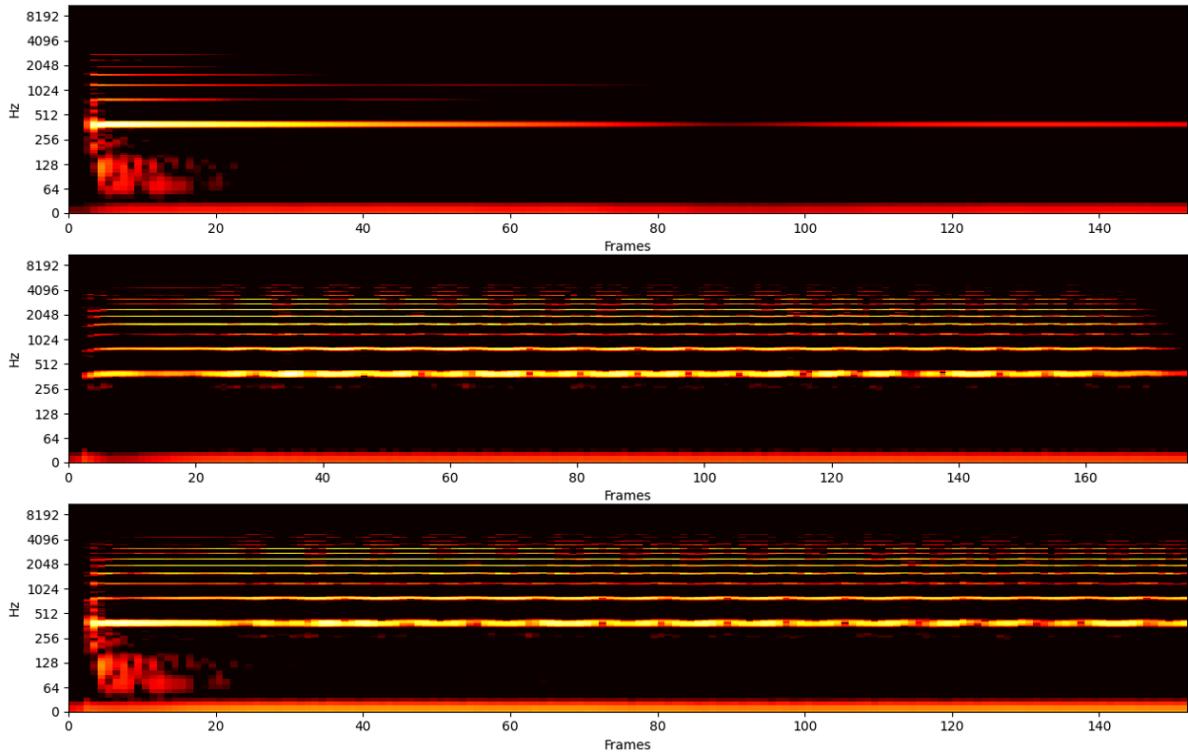


圖 2.1: 小提琴與鋼琴演奏 G4 音高三秒的特徵圖 上：小提琴，中：鋼琴，下：混合

雖然人耳可以輕易分辨出不同的音色，但是當兩種聲音混合在一起時，要精準的將兩種樂器的聲音分開是一件困難的任務，圖 2.1 的下方圖顯示了音訊混合後的特徵長相，若要將兩種樂器的音源分離，我們必須精準地知道每個頻率原本的成分才有可能乾淨的分離。音源分離的難度與樂器音色的複雜度、樂器種類的相似度而提高，因此至今音源分離的任務還是非常具有挑戰性。

另外在音樂追蹤方面，使用兩段相似的音訊特徵來追蹤是常見的方法，特徵越相近越容易有好的追蹤結果，但若是屬於音色變化較大的樂器（例如小提琴），儘管是同一人演奏相同曲目兩次，特徵也不一定相同，圖 2.2 顯示同一人使用相同速度、曲目演奏兩次的特徵圖，雖然大致上看起來都相同，但細看的話在時間軸 = 100 frames 時兩邊的特徵還是有些微的不同，這些些微的不同之處都有可能會影響到追蹤結果。圖 2.3 顯示不同人演奏同一首曲子的特徵圖，可以看到這兩段音訊明顯在環境的收音上可能差很多，且在曲子的詮釋上也不完全相同，使用這兩段音訊來追

蹤，難度會比同一人演奏的版本要高很多，因此使用不同音訊特徵來追蹤是具有一定的挑戰性。

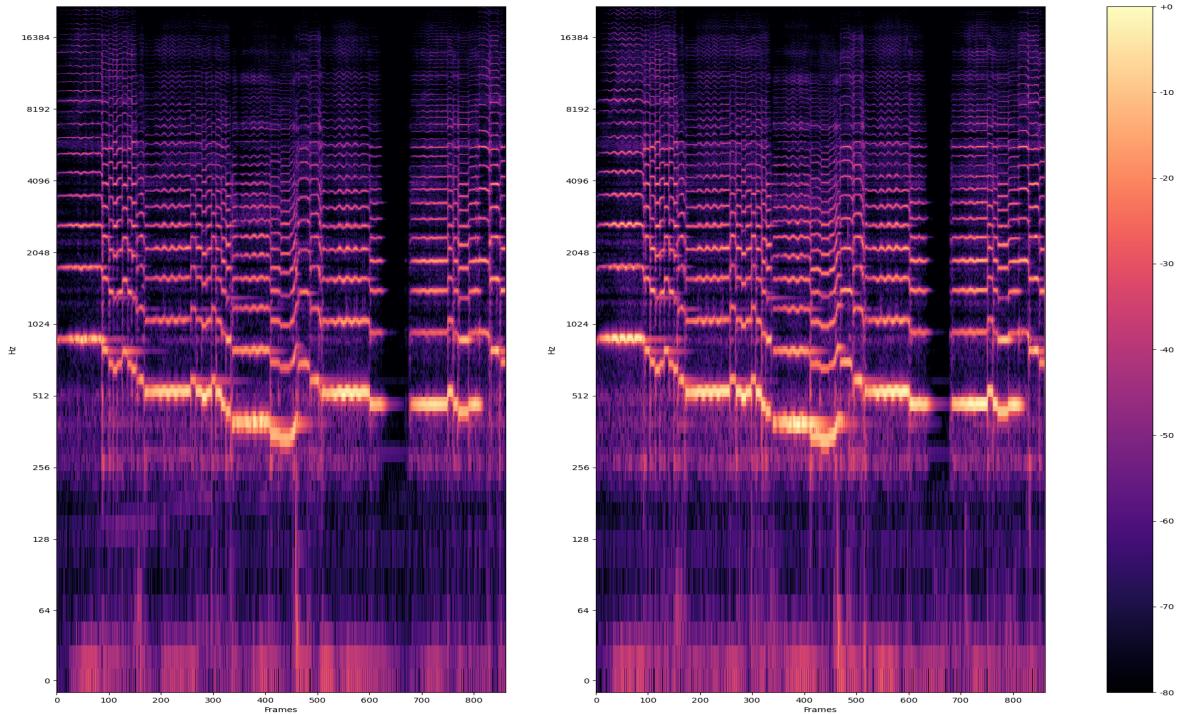


圖 2.2: 同一人演奏兩次 Beethoven Spring Sonata No.1 之特徵圖

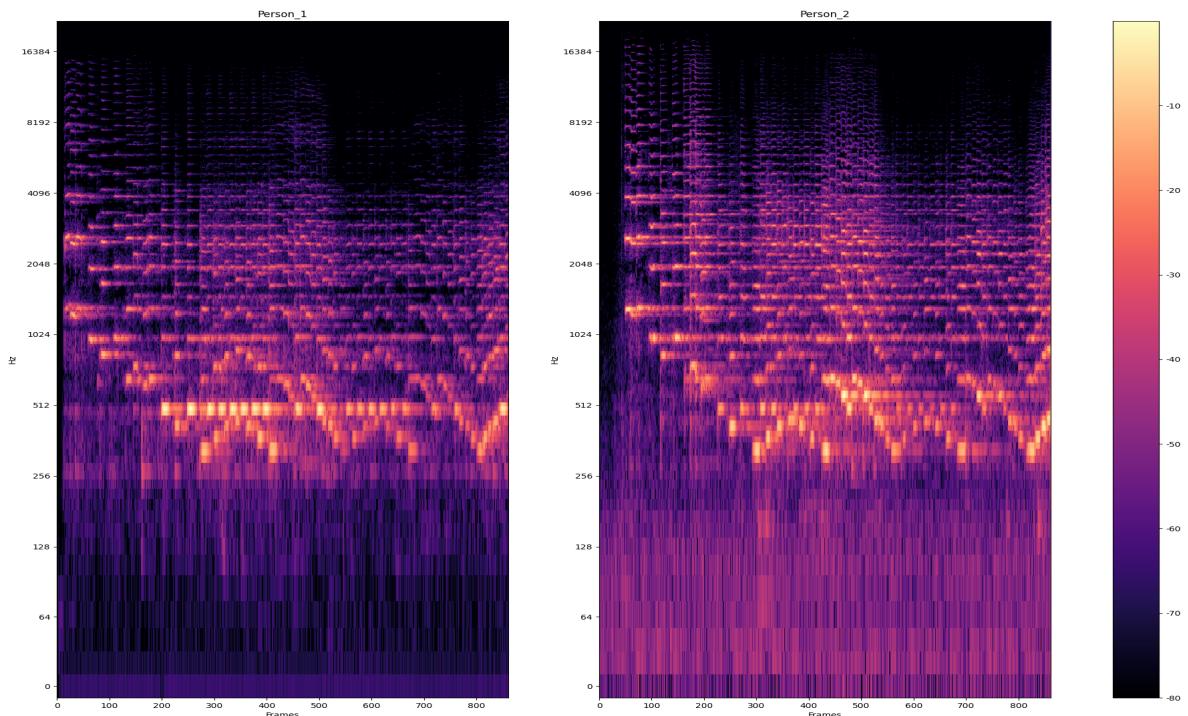


圖 2.3: 不同人演奏 Bach BWV1006 之特徵圖

2.2 文獻回顧

2.2.1 音源分離相關研究

音源分離為 MIR 中的一個重要領域，日常生活中我們所聽到的音樂大多數是來自多個音源的混合音訊，許多人可能希望調整混合音訊中的某種樂器的聲音平衡、改變聲音的空間位置，甚至將混合音訊與每個存在於其中的音樂來源作為其他 MIR 任務的訓練資料，因此，若是可以取得每個音樂來源的單獨音訊，上述所說的應用都能夠實現。

音源分離的目標最常見的為分離一首包含人聲、貝斯、鼓與其他(非前三者的所有音訊)的歌曲，在大型音源分離比賽 [24], [25] 中也是使用此標準進行。音源分離的方法可分為基於時間頻率表示方法與基於波形表示方法，第一種通常會預測每個來源的功率頻譜，並結合輸入的混合音訊相位來合成每個音源的波形，較傳統的方法有獨立成分分析(ICA) [30]，此方法的目標是找到一個分離矩陣，使混合訊號透過分離矩陣計算出獨立的來源訊號；非負矩陣分解 (NMF) [31] 為一種非監督式的機器學習技術，此方法是希望找到最好的兩個矩陣，使他們的乘積能接近原始的輸入。然而這兩種方法對於複雜的混合音訊通常無法完全恢復單個音訊。近年來隨著深度學習領域的蓬勃發展，越來越多研究使用全連接神經網路、LSTM、RNN 等神經網路訓練頻譜圖。2018 年由 Stöter 等人所提出的 Open-Unmix [32] 模型在當時在公開資料集 MUSDB [33] 上表現最佳，此模型使用三層雙向 LSTM 對輸入預測一個遮罩，將混合音訊透過遮罩得到分離音源頻譜圖。隨後在 2022 年由 Luo Yi 等人所提出的 Band-Split RNN [34] 在 MUSDB 上取得最佳的 SDR 成績，此模型將輸入的頻譜圖分割為預先定義好頻帶寬度的子帶頻譜圖，將子帶頻譜圖轉為相同維度的特徵後利用堆疊 RNN 交錯訓練，最後經過 MLP 生成遮罩並與混合輸入結合產生分離音源頻譜圖，至今仍是基於時間頻率訓練的開源模型中效果最佳的模型。Open-Unmix 與 Band-Split RNN 皆允許使用

者自訂不同的分離目標，使的這兩個模型擁有很多靈活性。

第二種基於波形訓練的方法最早提出的模型為 Wave-U-Net [35]，近年來較多人使用的 Demucs [3] 就是基於 Wave-U-Net 的架構開發的，Demucs 為 Encoder/Decoder 架構，內部由卷積 Encoder/Decoder 與雙向 LSTM 組成，Encoder 與 Decoder 使用 Skip U-net 連接，將混合立體音訊作為輸入，輸出每個來源的估計波型。2021 年 MDX 競賽 [24] 由 Meta AI 提出的冠軍模型 Hybrid Demucs [36] 更是將 Demucs 擴展為多域分析的架構，模型變為由波型、頻譜和共享層組成。2022 年 Meta AI 結合先前的結果提出了 Hybrid Transformers [37]，將 Encoder/Decoder 內部兩層替換為跨域 Transformer，使的模型可以接受異質資料，變成更靈活的架構，並在 MUSDB 的 SDR 表現上為 9.20dB，為目前所有開源模型中最佳的模型。

在古典樂器錄音的音源分離研究中，比起音高樂器（例如：吉他）與非音高樂器（例如：鼓）的音源分離情況更有挑戰性，且公開資料集的蒐集難度高，訓練資料驅動模型的難度也更大。因此針對這些資料通常會使用資料增強的方法來增加訓練資料，由 Miron 等人提出的 CNN-based 低延遲單聲道源分離模型 [38]，透過改變速度、音色、動態時間局部變化來增加新的訓練資料，使模型在實際情況下更能應對演奏時的狀況。由 Chiu Ching Yu 等人提出針對資料規模小的資料集的資料增強方法 [39]，使用 Open-Unmix 模型訓練並在針對小提琴與鋼琴的混合音源分離的任務上，比當時可用的最佳模型 Spleeter [40] 的效果要好。

本研究將使用靈活性高且較輕量的 Band-Split RNN [34] 模型，針對小提琴與鋼琴的混合音源資料做訓練，旨在促進古典樂器錄音的音源分離研究進展。

2.2.2 音樂追蹤相關研究

音樂追蹤在 MIR 領域中是一項重要的應用，與節拍追蹤 [41]–[43]、樂譜追蹤 [44]、自動伴奏 [8] 等相關技術有密切的關係，本節專注探討以音訊特徵為主的即時音樂追蹤的方法與應用的相關研究。音樂追蹤的方法包含隱藏馬爾科夫模型 (HMM) [45]，HMM 透過樂譜中的狀態迭帶訓練目前追蹤的位置，但也因為需要大量標註過的資料，在即時表現方面較為遜色；動態時間規整 (DTW) [46], [47] 為最多人使用的方法之一，DTW 透過計算兩段相似序列的成本矩陣，回推最佳對齊點，由於音樂的連續性使的 DTW 計算結果準確率高，但也因為計算時間複雜度高而無法應用在即時音樂追蹤。因此許多人對 DTW 演算法改良了更快速的版本，例如平行化動態時間規整 (PDTW) [48]，PDTW 將兩個時間序列分割成多段的序列，每段序列各自計算 DTW，計算完畢後再將每段序列連接回整段，加速 DTW 的運算時間；線上動態時間規整 (ODTW) [49]–[51] 將原本 DTW 向後對齊的模式改為增量對齊，ODTW 不需要事先取得兩段序列的資訊，而是根據每個時間點計算出的路徑決定下一步的方向，因此 ODTW 在即時音訊的處理的情況會比前面提到的方法還要快速，更能達到即時音樂追蹤的效果。

音樂追蹤使用的音訊通常越相像越好，例如使用同一人事先演奏的音訊來追蹤同一人的現場演奏，但通常使用者並不會特意去錄製事先演奏的音訊，尤其是在沒有合奏者的情況下，因此本研究致力於研究不同特徵的即時音樂追蹤系統。本研究將基於 [51] 所提出的架構進行重新實現並改良部分方法，希望能藉此達到我們所期望的效果並促進即時音樂追蹤研究進展。

三、研究方法

3.1 系統架構

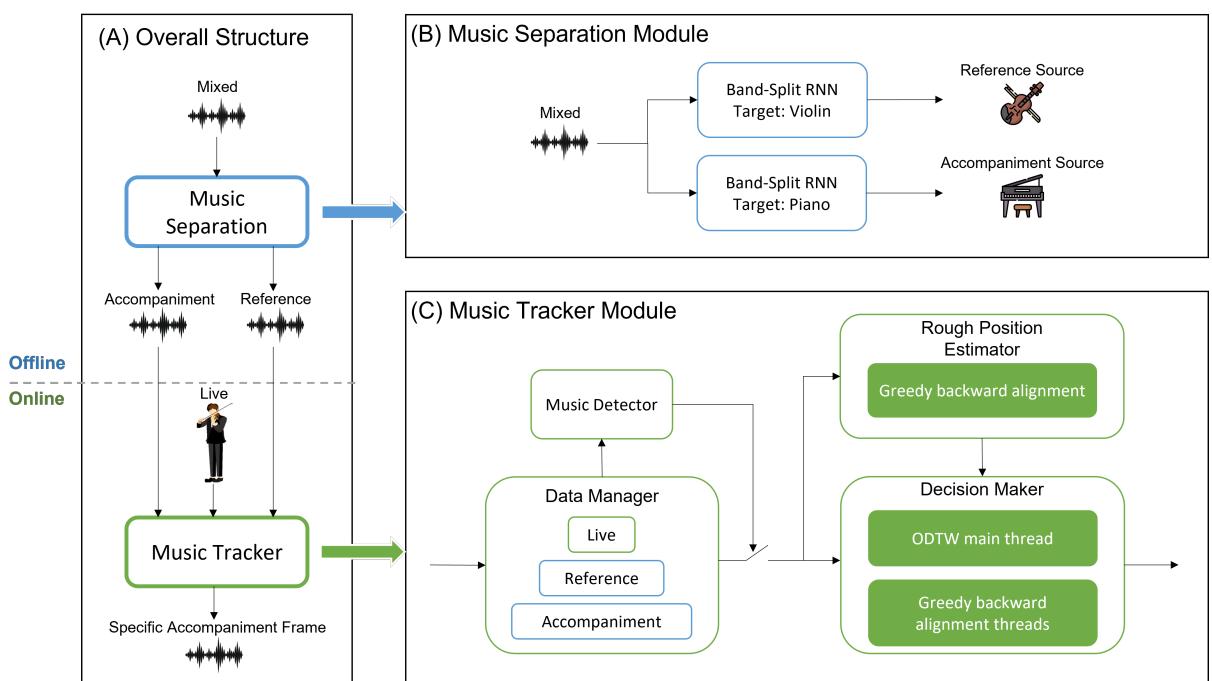


圖 3.1: 系統架構圖

圖 3.1 (A) 為本系統的整體架構，此系統可分為離線處理與線上處理兩個階段，離線處理階段會將混合音源分離成參考音訊與伴奏音訊作為線上處理階段的資料，線上處理階段接收演奏者的即時串流音訊，並結合參考音訊來計算目前演奏者的樂曲位置，並根據樂曲位置輸出對應的伴奏音訊。請注意本研究將小提琴與鋼琴之混合音訊作為主要資料，並專注在當演奏者樂器為小提琴或鋼琴的狀況下來使用本系統。

圖 3.1 (B) 為音源分離模組的細部架構，採用深度學習模型將混合音

訊中的目標音源分離出來。此模組包含了小提琴(參考音訊)音源分離模型與鋼琴(伴奏音訊)音源分離模型，另外在訓練階段，本研究設計了一種簡單的頻帶切割估計方法提升模型的分離效果。關於模型架構、頻帶切割估計方法會在3.2節做討論。

圖 3.1 (C) 為音樂追蹤模組的細部架構，此模組使用 Python 3.9 開發，我們使用 multiprocessing package [52] 實現平行化處理，串流音訊使用 pyaudio package [53] 接收處理，達到即時伴奏的效果。此模組包含四個元件，分別為 Data Manager、Music Detector、Rough Position Estimator 與 Decision Maker。Data Manager 管理所有音訊的資料與特徵、Music Detector 即時偵測音樂是否開始、Rough Position Estimator 粗略估計可能的伴奏時間位置，提供 Decision Maker 更多選擇、Decision Maker 即時計算目前輸出位置並決定最後的輸出位置，關於元件詳細的設計與改動會在3.3節做討論。

3.2 音源分離模組

3.2.1 Band-Split RNN

Band-Split RNN (BSRNN) [34] 由 Luo Yi 等人於 2022 年提出，是一種專門為高取樣率訊號所設計的時頻域音源分離模型。BSRNN 將輸入的頻譜圖 $X \in \mathbb{C}^{F \times T}$ 根據自定義的頻帶切割點 $G = (G_1, G_2, \dots, G_i)$, $\sum_{i=1}^K G_i = F$ 拆分成一個個的子帶頻譜圖 $B = (B_1, B_2, \dots, B_K)$, $B_K \in \mathbb{C}^{G_K \times T}$ ，每個不同大小的子帶頻譜圖經過處理後變為一系列具有相同維度的特徵 $Z \in \mathbb{R}^{N \times K \times T}$ ，接著利用 RNN 層將頻帶與序列的資訊交錯建模得到輸出 $Q \in \mathbb{R}^{N \times K \times T}$ ，最後每個子帶特徵由多層感知器 (MLP) 產生對應的複值時頻遮罩 $M \in \mathbb{C}^{F \times T}$ ，並將遮罩用於輸入的混合頻譜圖來產生估計的目標源頻譜圖 $S \in \mathbb{C}^{F \times T}$ 。整體架構如圖 3.2 所示。

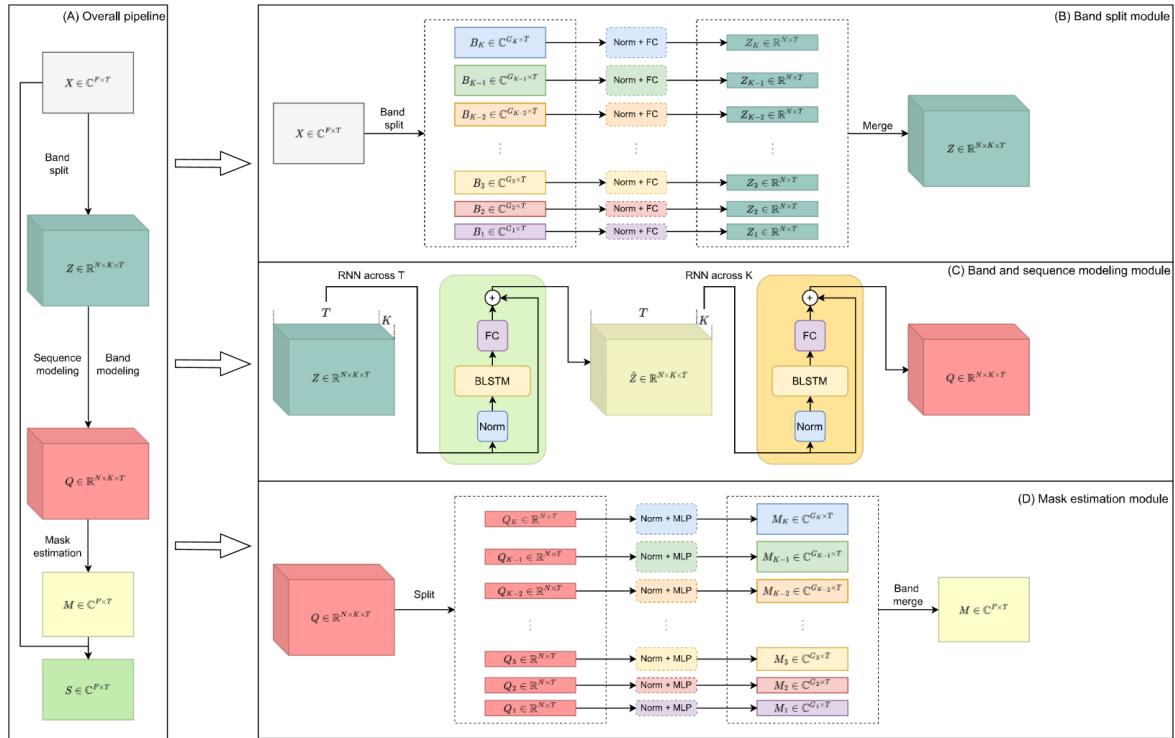


圖 3.2: BSRNN 架構圖，擷取自 [34]

其中在圖 3.2 (B) 的頻帶切割模組是 BSRNN 模型中最重要的模組，此模組可根據目標音源的聲音特性來設計不同的頻帶切割範圍，例如當目標音源的聲音特性都集中在較低頻的地方時，我們可以將低頻切割得更細，高頻切割的寬一點，使模型在訓練時可以看到更重要的頻域特徵。

3.2.2 估計不同樂器的頻帶切割點

為了使頻帶切割模組 (圖 3.2 (B)) 發揮更好的效果，我們使用簡單的頻帶切割估計方法來估計小提琴與鋼琴的頻帶切割點，此方法不需要依賴先驗知識或專家知識的幫助，而是透過同一種樂器的音訊資料來估計頻帶切割點。首先我們對每種樂器選取約 15 分鐘的音訊資料，這些資料並不會加入模型的訓練。接著定義每種樂器的切割比例閥值 r 、最小頻率切割值 $f_{min} = 100\text{Hz}$ 、取樣率 $sr = 44100\text{Hz}$ 與初始頻帶陣列 $F_{band} \in \frac{sr}{2}$ ，本研究之兩種樂器的切割比例閥值為 $r_{violin} = 0.013$, $r_{piano} = 0.0035$ ，此數值是根據計算出來的切割頻帶數量去做調整。

對於每筆資料，使用快速傅立葉轉換計算頻率域的每個樣本得到的振幅與對應到的頻率，由於 FFT 的對稱性，我們只需要迭帶一半的樣本數，並將振幅值累加到對應的 F_{band} 。累加振幅值越高代表此頻率的重要程度越高，如圖 3.3 所示。

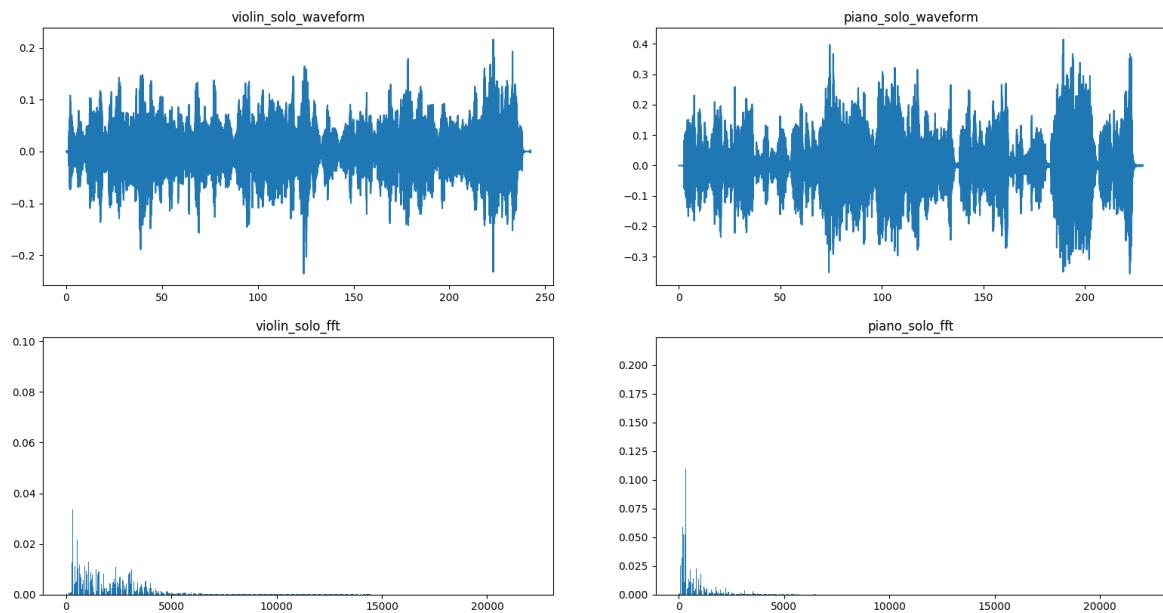


圖 3.3: 兩種資料的波形圖(上)與經過 FFT 轉換後的 F_{band} 圖(下)

最後我們對 F_{band} 的範圍迭帶計算每個區間的累積振幅值占總和值的比例 $f_{range} / \sum F_{band}$ ，若超過切割比例閥值 r ，則在此頻率點增加一個切割點。依照上述的切割比例我們得到兩種樂器的頻帶切割點為

Violin = [400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000, 3200, 3400, 3600, 3800, 4000, 4200, 4500, 4800, 5300, 5800, 6500, 7400, 8600, 10400, 13000, 18000]

Piano = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000, 3200, 3400, 3600, 3900, 4100, 4400, 4800, 5300, 5900, 6900, 8700, 17300]

在訓練 Band-Split RNN 模型時我們將會套用上面兩個頻帶切割範圍做訓練。

3.3 音樂追蹤模組

此模組是參考 [51] 的 real-time music tracker 來重新實現，本節將分成兩個部分說明，3.3.1節至3.3.3節會介紹實現音樂追蹤模組所使用到的演算法並詳細說明改進的部分，3.3.4節至3.3.7 會介紹每個元件的設計與改動。

3.3.1 動態時間規整演算法

動態時間規整演算法 (DTW) [54]，是一種用來計算兩個時間序列資料之間相似度的方法。給定兩個時間序列 $X = (x_1, x_2, \dots, x_N), N \in \mathbb{N}$ 和 $Y = (y_1, y_2, \dots, y_M), M \in \mathbb{M}$ ，其中 N 和 M 為兩個時間序列的長度， x_N 和 y_M 是任兩個具有時間順序且形狀相似的資料，例如音樂、語音等等，如圖 3.4 所示。

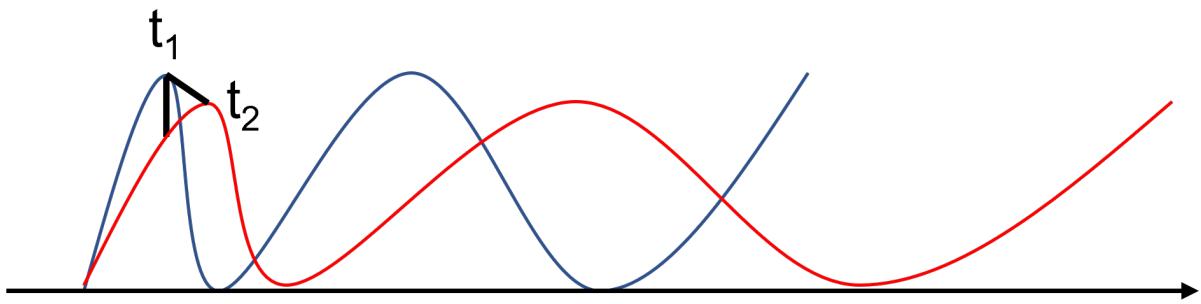


圖 3.4: 兩個相似的時間序列：藍色: X 紅色: Y

可以看到計算 $X[t_1]$ 與 $Y[t_1]$ 的距離，並不會是相似的資料點，但若計算 $X[t_1]$ 與 $Y[t_2]$ 的距離，就會被認為是兩個相近的資料點，因此當兩段相似的資料但在時間軸上並不是對齊的，就適合用 DTW 演算法動態對齊時間。考慮到時間序列的長度不一定相等的情況，例如不同人演奏同一個音符可能會有不同的速度差異，因此 DTW 通常採用動態規劃的方法來計算相似度。

首先使用歐基里德距離式 (3.1) 計算 X 與 Y 所有時間點的距離，

$$d(i, j) = \sqrt{|x_i - y_j|^2} \quad (3.1)$$

並根據式 (3.2) 使用動態規劃方式建立距離矩陣 $D \in \mathbb{R}^{N \times M}$ 。

$$\left\{ \begin{array}{l} D(0, 0) = d(x_1, y_1) \\ D(i, 0) = D(i - 1, 0) + d(x_i, y_1) \\ D(0, j) = D(0, j - 1) + d(x_1, y_j) \\ D(i, j) = d(x_i, y_j) + \min \left\{ \begin{array}{l} D(i, j - 1) \\ D(i - 1, j) \\ D(i - 1, j - 1) \end{array} \right\} \end{array} \right. \quad (3.2)$$

在計算最短路徑時，必須遵守三個限制：

1. 邊界限制：路徑的開頭與結束必須為 (x_1, y_1) 與 (x_N, y_M)
2. 連續性：若路徑的上一個對齊點為 (i, j) ，下一個對齊點只能是 $(i + 1, j + 1), (i + 1, j), (i, j + 1)$
3. 單調性：若路徑的上一個對齊點為 (i, j) ，下一個對齊點必須為 $(i_{next}, j_{next}) \in (i_{next} \geq i, j_{next} \geq j)$

這是為了使回溯的路徑為連續無倒退且整段序列都被包含在輸出裡，確保對齊的一致性。

最後從 $D(N, M)$ 往前回溯至 $D(1, 1)$ 尋找一條最小的累積距離成本路徑，計算公式如式 (3.3) 所示，選擇回溯點時會確認行、列與對角三個

方向的距離成本值，並選擇成本最小的點作為回溯點，

$$\begin{cases} (i_{next}, j_{next}) = \operatorname{argmin}_{(i-1,j-1), (i-1,j), (i,j-1)} D(i_{next}, j_{next}) \\ \text{Stop, } \quad \text{if } (i, j) = (1, 1) \end{cases} \quad (3.3)$$

當回溯點走到起始點 $(1, 1)$ 時，將所有回溯點連起來就得到最後對齊路徑 W ，如示意圖 3.5 所示，右上角為路徑的結束位置 $D(N, M)$ ，左下角為路徑的開始位置 $D(1, 1)$ ，綠色路徑即為 W 。

10	10	11	18	17
9	11	13	12	11
9	11	13	6	8
2	4	5	8	12
1	2	3	10	16

圖 3.5: DTW 累積距離矩陣回溯路徑示意圖

本研究所使用的 DTW 演算法來自 python librosa package [55] 中的 librosa.sequence.dtw 函式。

3.3.2 線上動態時間規整演算法

由於 DTW 演算法在時間複雜度上為 $O(NM)$ ，屬於複雜度高的演算法，因此當某個序列的時間較長或是不確定序列的時間長度（例如即時串流音訊）時，DTW 演算法在計算就會花費許多時間。

線上動態時間規整演算法 (ODTW) [49] 由 Dixon 等人提出，是基於 DTW 演算法所做的改良，首先 ODTW 在計算對齊路徑時並不是回溯計算而是向前增量計算，並且將計算距離成本的範圍縮小至預先定義的範

圍 c ，因此使 ODTW 在時間與空間複雜度的表現都是線性 $O(N)$ ，且因為不需事先知道整個序列的資料，因此適合處理長序列音訊或即時串流音訊。

本研究實現 ODTW 演算法的步驟如下，給定兩個時間序列 $X = (x_1, x_2, \dots, x_N), N \in \mathbb{N}$ 和 $Y = (y_1, y_2, \dots, y_M), M \in \mathbb{M}$ ，其中 X 為現場音訊，因此 N 會隨著串流時間長度而增加，直到串流結束， Y 為事先給定的參考音訊，用於對齊現場音訊。我們想透過距離成本矩陣 D_{total} 找到 X 和 Y 的對齊路徑 $W = (W_1, W_2, \dots, W_k)$ ，其中每個 W_k 為有順序的 (i_k, j_k) 且 $(i, j) \in W$ 代表兩段序列在 x_i 與 y_j 點上對齊。

距離成本矩陣 D_{total} 透過 3.3.1 節所提到的式 (3.1) 與 式 (3.2) 計算，另外我們使用 D_{cell} 紀錄單格的距離成本 d 。我們定義搜尋範圍 $c = 450$ 來限制 D_{total} 與 D_{cell} 的大小，當有新的 i 加入時，我們計算 $d(i, j - c), d(i, j - c + 1) \dots d(i, j + c)$ 個點，此時 j 為目前演算法計算到的參考音訊時間點，而對於 j 點，則是計算 $d(i - c, j), d(i - c + 1, j) \dots d(i, j)$ ，由於我們無法得知未來的現場音訊，所以只能計算到目前最新的時間點 i 。上述的計算方式與傳統的 ODTW 演算法有些微的不同，在 [49] 中，每次迭帶時在 (i, j) 方向計算的格數皆為 c 個，我們將每次迭帶在 j 方向計算的 cell 個數改為 $2c - 1$ ，如範例圖 3.6 所示。

會做這樣的更動是因為我們的 i 為現場串流音訊的時間點，而串流音訊並不會出現相同的時間點 (輸出永遠為嚴格單調遞增)，且我們無法得知目前現場音訊的快慢，因此為了能讓最後輸出對齊位置時有更多的選擇，我們除了向前計算，也向後計算參考音訊的距離成本。

在計算完目前 (i, j) 的距離成本矩陣後，我們需要決定下一步的 (i_{next}, j_{next}) 走向以增加新的距離成本到兩個矩陣中。由於我們將 i 設定為嚴格單調遞增，因此在決定增量對齊的方向時，只會有 $(i + 1, j + 1)$ 與 $(i + 1, j)$ 兩種方向。除非在同一個方向增加超過事先設置的參數 $ODTW_{Max_Run} = 3$ ，否則我們每次都會計算 $(i + 1, j + 1)$ 方向。另外當目前的 $i < c$ 時，為了使搜索範圍內的格數有計算到完整的距離成本，即

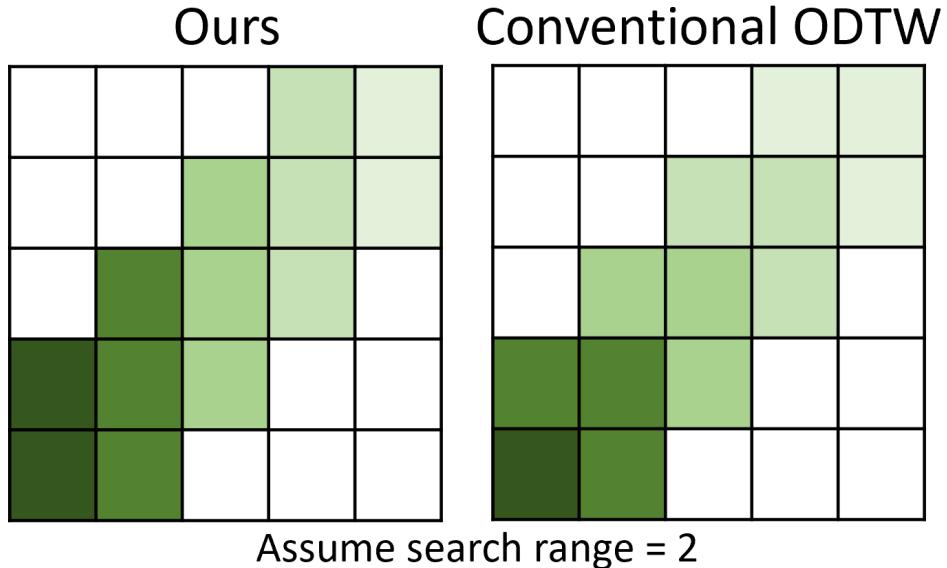


圖 3.6: 每個迭帶所計算到的距離成本矩陣格數位置比較

使 $output_j \neq j$ ，在下一個迭帶我們還是計算點 $(i + 1, j + 1)$ ，直到 $i >= c$ 時才會將 j 設為 $output_j$ 。

在每次計算完最新的 D_{total} 與 D_{cell} 後，我們使用這兩個成本矩陣計算時間點 i 的對齊位置 $output_j$ ，首先我們設定兩個參數 $L_{upper} = (j + ODTW_{Max_Run} \times 10)$, $L_{lower} = (j - 1)$ ，分別代表尋找 $output_j$ 的上下限，迭帶所有介於 L_{upper} 與 L_{lower} 的點 $curr_j$ ，找到一組點 $(i, curr_j)$ 的 $D_{total}(i, curr_j) + D_{cell}(i, curr_j)$ 值為最小值， $(i, curr_j)$ 即為 ODTW 的輸出對齊點。

3.3.3 貪心向後對齊方法

由於 ODTW 在對齊時為增量對齊，並不像 DTW 一樣在計算完兩段音訊的成本矩陣後才回溯最佳對齊路徑，因此通常 ODTW 的輸出結果會比 DTW 的輸出結果來的更不穩定。為了改善上述問題，我們參考 [50] 中的 Greedy Local Search 演算法來設計 Greedy Backward Alignment (GBA) 方法，此方法配合多執行續的執行可以快速的計算目前位置的向後對齊成本。

給定一個預計輸出位置 (i, j) 作為向後對齊的結尾點，此位置代表在現場音訊為 i 時，參考音訊估計的對齊位置為 j ，我們設定一個向後對齊距離參數 GBA_t ，根據現在 i 的時間點與先前輸出位置紀錄 W_{output} ，我們可以推算出向後對齊的開頭點 $(i_{prev}, j_{prev}) = W_{output}[i - GBA_t]$ 。

接著根據距離成本矩陣 D ，我們可以透過3.3.1節的式 (3.3) 計算 (i_{prev}, j_{prev}) 到 (i, j) 的對齊成本 W_{cost} ，在回溯的過程中，因為 (i_{prev}, j_{prev}) 與 (i, j) 並不是真正的起始與結束點，我們設定了最大步數 $GBA_{Max_Run} = 3$ 來限制回溯路徑的走向，若過程中在行或列的方向走超過 GBA_{Max_Run} ，下一步只計算其他兩種方向的對齊成本決定走向。另外在計算累積距離成本時，我們紀錄回溯路徑的總步數 $step_{total}$ ，並針對三種走向設計不同大小的值，當走向為對角時， $step_{total}$ 增加 2 步，當走向為行或列時， $step_{total}$ 增加 1 步。這是為了正規化總累積成本，使每個預計輸出位置在比較累積距離成本時能公平的比較，最後對齊成本計算公式如式 (3.4) 所示。

$$W_{cost} = \frac{\sum_{w=(i_{prev}, j_{prev})}^{(i, j)} D(w)}{step_{total}} \quad (3.4)$$

3.3.4 Data Manager Block

在線上處理階段啟動前與啟動後，我們對非串流音訊（參考音訊、伴奏音訊）與串流音訊（現場音訊）擷取兩種特徵作為其他元件的輸入資料，分別為高解析度特徵與低解析度特徵，高解析度特徵為梅爾頻譜特徵，低解析度特徵為高解析度特徵對時間軸卷積後的重新採樣。為了管理音訊特徵的處理與傳送，我們設計了 Data Manager 來集中管理音樂追蹤模組中所有用到的音訊特徵。

在啟動追蹤系統前，Data Manager 會拿到音源分離模組分離出來的參考音訊與伴奏音訊，對於這兩段完整的音訊，使用 44100Hz 的取樣率讀取，並計算 STFT 得到頻譜圖，STFT 的參數設定如表 3.1 所示。

為了使特徵更接近人類的聽覺系統，我們對 STFT 頻譜圖做梅爾頻

表 3.1: STFT 參數設定

Window size	46ms (~ 2029 frames)
Hop size	20ms (~ 882 frames)
Number of fft	46ms (~ 2029 frames)
Window function	Hanning

譜 (Mel spectrogram) 分析，得到高解析度特徵。梅爾頻譜是根據人耳對聲音的敏感度所設計的一種時頻分析方式，而梅爾頻譜就是由梅爾濾波器組 (Mel filter bank) 與 STFT 頻譜圖矩陣相乘後得到的結果，梅爾頻譜圖的參數設定如下表 3.2 所示， n_{mel} 為濾波器數量， f_{min} 與 f_{max} 為使用率波器的頻率範圍，代表梅爾頻譜特徵在 0-8000 Hz 使用了 84 個濾波器將此頻率範圍壓縮成更接近人耳所聽到的樣子。

表 3.2: 高解析度特徵參數設定

n_{mels}	84
Number of fft	46ms (~ 2029 frames)
f_{min}	0 Hz
f_{max}	8000 Hz

低解析度特徵由前面所提到的高解析度特徵對時間軸進行卷積，參數設定如表 3.3 所示，將每個 window size 的高解析度特徵窗與 window function 做卷積並取平均值代表此低解析度特徵時間點的頻率份量。

表 3.3: 低解析度特徵參數設定

Window size	600ms (~ 30 frames)
Hop size	300ms (~ 15 frames)
Window function	Hanning

追蹤系統啟動後，音訊以單聲道、每次傳送樣本數 = 2048 個傳送串流音訊到系統，對於串流音訊的特徵處理方法與上述相同，但由於在處理特徵時我們有設定特徵的 window size，因此必須確保接收到足夠的樣本數才能擷取特徵，否則可能導致特徵蒐集不完全而失真。因此當系統啟動後，我們將現場音訊記錄下來，當音訊累積足夠的長度時再擷取高/低解析度特徵。

圖 3.7為現場音訊與參考音訊之特徵圖，可以看到現場音訊雖然是以 frame 為單位處理特徵，但將特徵合併後並沒有任何雜訊，另外可以看到高解析度特徵的時間軸切片較細，低解析度特徵的時間軸切片較粗。

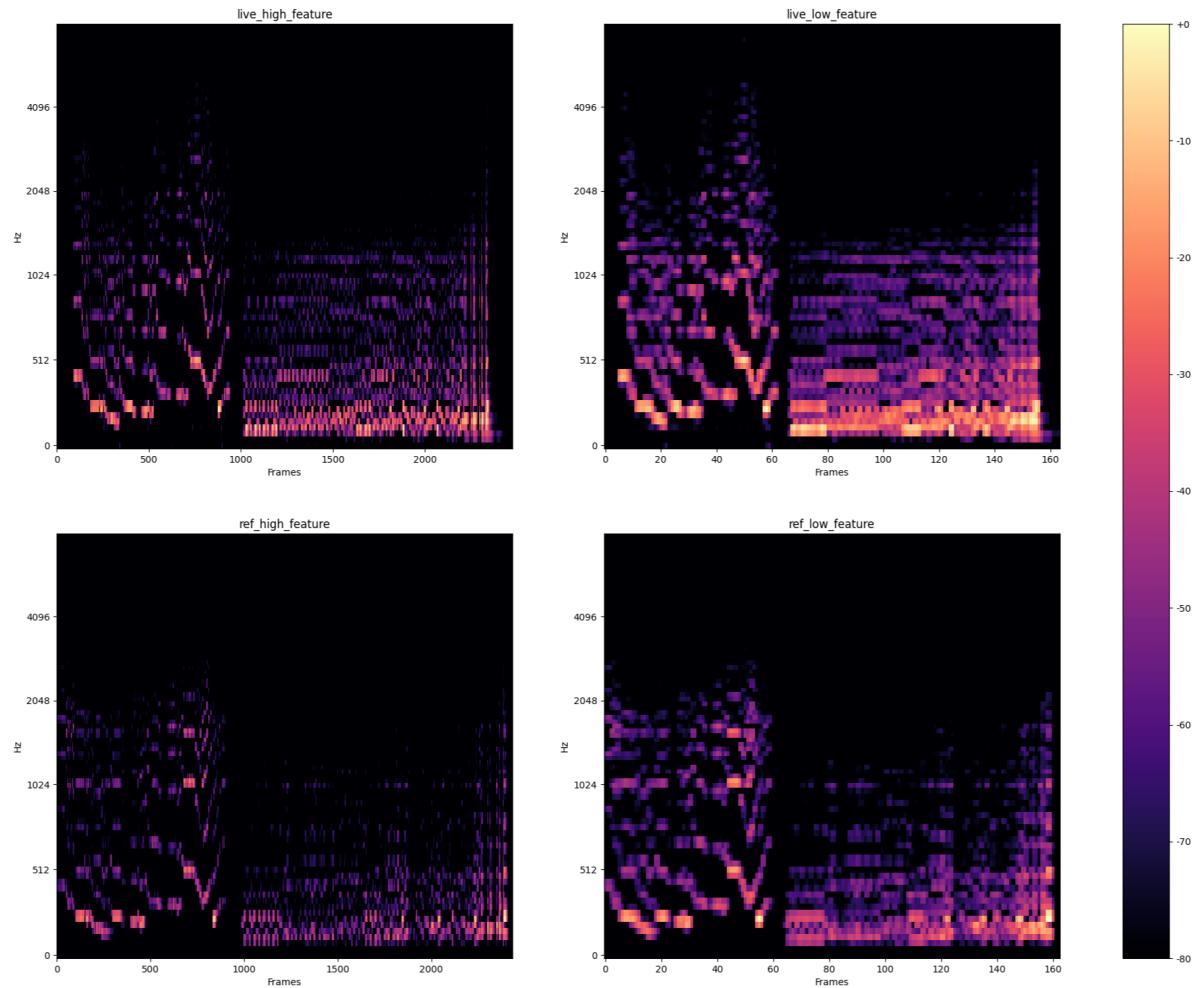


圖 3.7: 上：現場音訊的高(左)/低(右)解析度特徵圖；下：參考音訊的高(左)/低(右)解析度特徵圖

3.3.5 Music Detector Block

Music Detector 為負責偵測使用者開始演奏的時間並觸發後面元件啟動的 process，我們實作了平均振幅的閥值判斷並參考 [51] 的 DTW 對齊成本閥值判斷方法來實現。

當 Music Detector 接收到音訊時，我們計算平均振幅並根據事先定義

的閥值 θ_A 判斷此片段是否為靜音片段，振幅平均值的計算公式如式 (3.5) 所示， N 為整段音訊的樣本數， $|x[n]|$ 為音訊第 n 個樣本的絕對值， A 為振幅平均值。

$$A = \frac{1}{N} \sum_{n=1}^N |x[n]| \quad (3.5)$$

θ_A 的值是透過統計的方式來決定設定的值，我們使用已標記音樂開始時間的音訊來一一測試，已標記的音訊是使用自己錄製的音檔來測試，我們透過測量每一首的音樂開始時間的振幅值與音樂尚未開始時間的振幅值，並將音樂開始時間的振幅值加總後取平均獲得，最後我們將 θ_A 設為 0.01。

式 (3.6) 為平均振幅與閥值的條件判斷公式，當 $A \leq \theta_A$ 時，代表此片段為靜音片段，因此直接回頭計算新的音訊的平均振幅，當 $A > \theta_A$ 時，代表此片段為非靜音片段，可能為音樂開始的地方，因此將此片段接著使用 DTW 對齊成本閥值判斷方法。

$$\begin{cases} \text{Compute DTW cost,} & \text{if } A > \theta_A \\ \text{Do nothing,} & \text{if } A \leq \theta_A \end{cases} \quad (3.6)$$

DTW 對齊成本閥值判斷方法我們參考 [51]，當現場音訊通過平均振幅閥值的判斷後，我們將音訊擷取為高解析度特徵並與參考音訊開頭 0.5 秒的高解析度特徵計算 DTW 對齊成本 DTW_{cost} 。若 DTW_{cost} 小於設定好的閥值 θ_{cost} ，Music Detector 就會觸發 tracking event。 θ_{cost} 的值與上述提到 θ_A 的設定方法一樣為統計的方式，我們將 θ_{cost} 設為 2000。

式 (3.7) 為 DTW 對齊成本與閥值的條件判斷公式，當 $DTW_{cost} \geq \theta_{cost}$ 時，認定此兩段音訊的相似度不足，因此不觸發後面的元件並回頭計算新的音訊的平均振幅，當 $DTW_{cost} < \theta_{cost}$ 時，認定此兩段音訊具有足夠的相似度被認定為音樂開始的片段，因此觸發後面的元件並關閉自

己的 process。

$$\begin{cases} \text{Trigger tracking event,} & \text{if } DTW_{cost} < \theta_{cost} \\ \text{Do nothing,} & \text{if } DTW_{cost} \geq \theta_{cost} \end{cases} \quad (3.7)$$

3.3.6 Rough Position Estimator Block

在追蹤系統啟動後，為了避免 Decision Maker 的錯誤對齊，Rough Position Estimator (RPE) 負責使用低解析度特徵計算目前最新現場音訊幀與參考音訊每一幀的相似度，並挑出相似度高的前 8 個粗略估計位置提供給 Decision Maker 參考。

當新的音訊幀經過特徵擷取後，RPE 會獲得新的現場低解析度特徵 $Live_{low}[t] \in \mathbb{R}^{1 \times F}$ ， t 為最新的低解析度特徵時間點。此時 RPE 計算 $Live_{low}[t]$ 與整個參考低解析度特徵 $Ref_{low} \in \mathbb{R}^{N \times F}$ 的歐基里德距離並紀錄到距離成本矩陣 $D_{RPE} \in \mathbb{R}^{t \times N}$ ， N 為 Ref_{low} 的時間長度。式 (3.8) 表示在現場與參考的時間點為 $[t, n]$ 時兩個低解析度特徵的距離。圖 3.8 為實際計算出來的 D_{RPE} ，越亮的地方代表兩格特徵距離越遠。

$$D_{RPE}[t, n] = \sum_{f=0}^F \sqrt{|Ref_{low}[n, f] - Live_{low}[t, f]|^2} \quad (3.8)$$

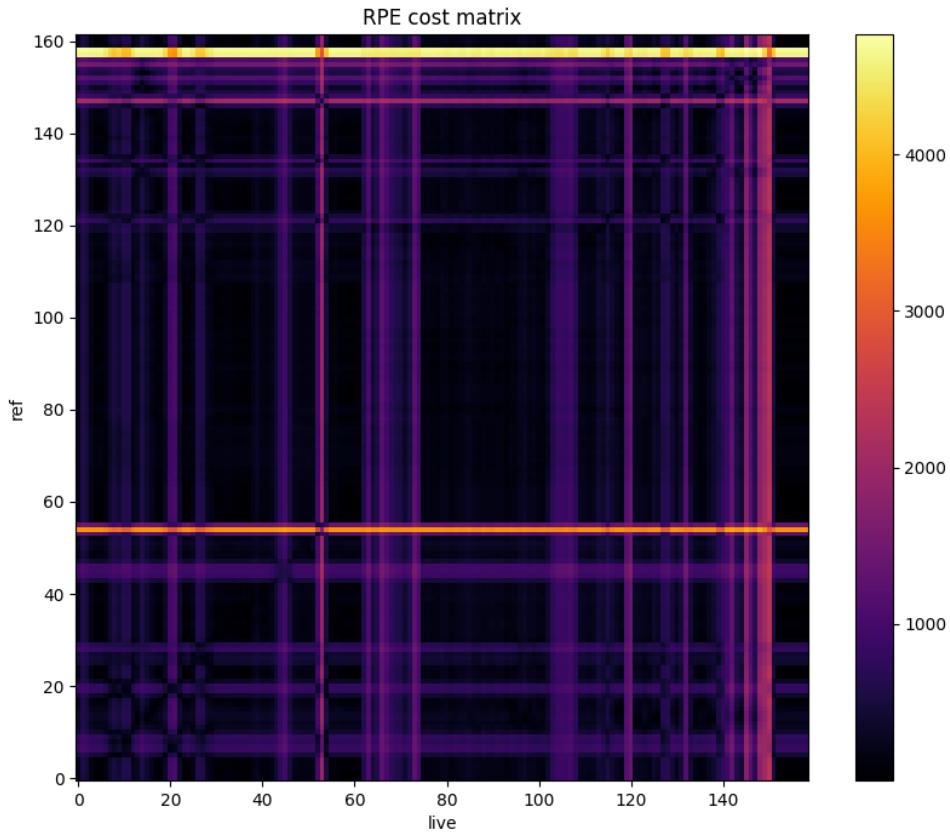


圖 3.8: RPE 距離矩陣

當 D_{RPE} 累積現場 9 秒 (~ 30 low feature frames) 的資訊後，使用 GBA 方法計算時間點 t 時 Ref_{low} 所有時間點向後對齊 9 秒的最小路徑成本矩陣 $C_{RPE} \in \mathbb{R}^{1 \times N}$ ，我們將 RPE 的 GBA_{Max_Run} 設定為 3，當最小路徑成本值越小，代表此參考點與目前現場位置的相似度越高，因此我們對 C_{RPE} 做 Min-Max 資料正規化，公式如式 (3.9) 所示，所有最小成本路徑值都被正規化到 $[0, 1]$ 之間，並且相似度高的值越接近 1。

$$S_{RPE} = 1 - \frac{C_{RPE}[n] - \min(C_{RPE})}{\max(C_{RPE}) - \min(C_{RPE})}, n = \{0, 1 \dots N\} \quad (3.9)$$

最後我們獲得相似度矩陣 $S_{RPE} \in \mathbb{R}^{1 \times N}$ ，為了取出前 8 個相似度最高的估計位置，我們對 S_{RPE} 進行降冪排序，並且設定了一個相似度閥值 θ_S ，當 S_{RPE} 中的值超過 θ_S 且排序在前 8 個順序時，才可以做為粗

略估計位置輸出給 Decision Maker，我們參考 [50] 的設定，將為 θ_S 設為 0.95。另外累積超過 9 秒的現場音訊後，下一次的 GBA 方法計算會在 1.5 秒 (~ 5 low feature frames) 後，這是為了節省計算資源，使系統能保持在即時的狀態。RPE 的粗略估計位置點如圖 3.9 的白色點所示。

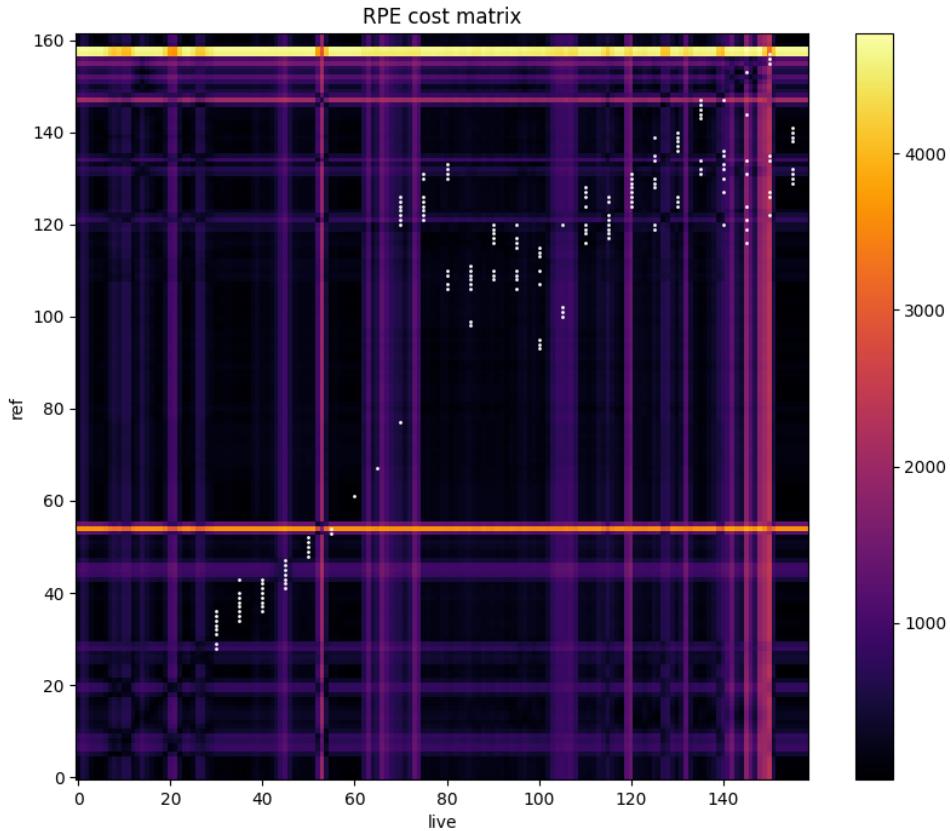


圖 3.9: RPE 粗略估計位置

3.3.7 Decision Maker Block

Decision Maker 由 ODTW thread 與 GBA threads 所組成，負責即時計算現場高解析度特徵與參考高解析度特徵的即時對齊位置，並參考 RPE 的粗略估計位置決定最後輸出，為此音樂追蹤模組中最重要的區塊。

在 ODTW thread 中會持續接收現場高解析度特徵 $Live_{high}[t] \in \mathbb{R}^{1 \times F}$ ，並與參考高解析度特徵 $Ref_{high} \in \mathbb{R}^{N \times F}$ 計算兩種距離矩陣 $D_{cell} \in \mathbb{R}^{t \times N}$

與 $D_{total} \in \mathbb{R}^{t \times N}$ ，ODTW 的計算方式與參數設定如3.3.2節所示，若目前沒有 RPE 的粗略估計位置加入，則輸出 ODTW 所計算出來的對齊位置。

GBA threads 會接收來自 RPE 所計算出來的粗略估計位置並使用 ODTW 的距離成本矩陣 D_{total} 計算 GBA 結果，為了不讓 RPE 完全控制輸出結果，我們將目前 ODTW 計算出來的輸出位置 $output_j$ 與 $(output_j \pm 15, 30, 45)$ 一併加入 GBA threads 計算最終輸出位置。我們將 GBA_t 設為 15 秒 (~ 750 high feature frames)，每個預計輸出位置將會向後計算 15 秒的最小路徑成本並選出成本最小的位置作為最後輸出。

圖 3.10為 ODTW 的累積距離成本圖與 Decision Maker 的輸出路徑，我們也將 RPE 的粗略估計位置 (黃色點) 結果與 ODTW 加入預計輸出的位置 (桃紅色點) 顯示在上面。為了瞭解 Decision Maker 輸出的路徑是否有偏差，我們將 DTW 的輸出路徑 (淺藍) 呈現在 圖 3.11，並將 Decision Maker 輸出路徑 (綠色) 疊加在上面，可以看到幾乎是與 DTW 的路徑重疊，代表此系統在即時對齊的能力與 DTW 離線對齊的能力相當。

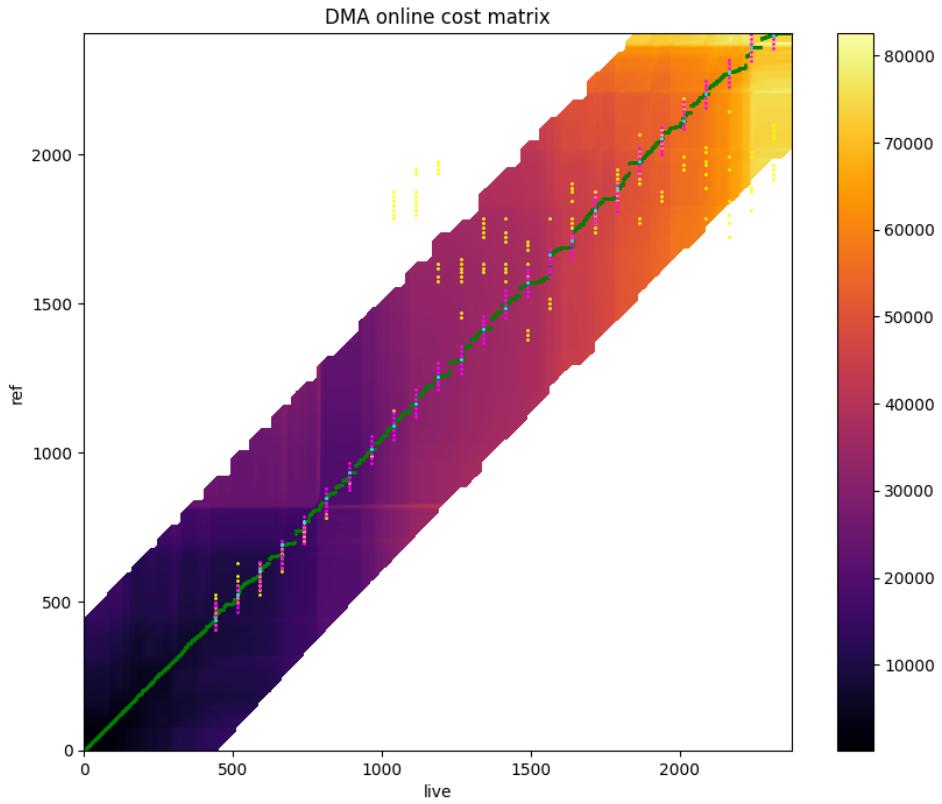


圖 3.10: Decision Maker 計算的輸出點：淺藍：GBA 計算的最後輸出位置 綠色：ODTW 計算的最後輸出位置

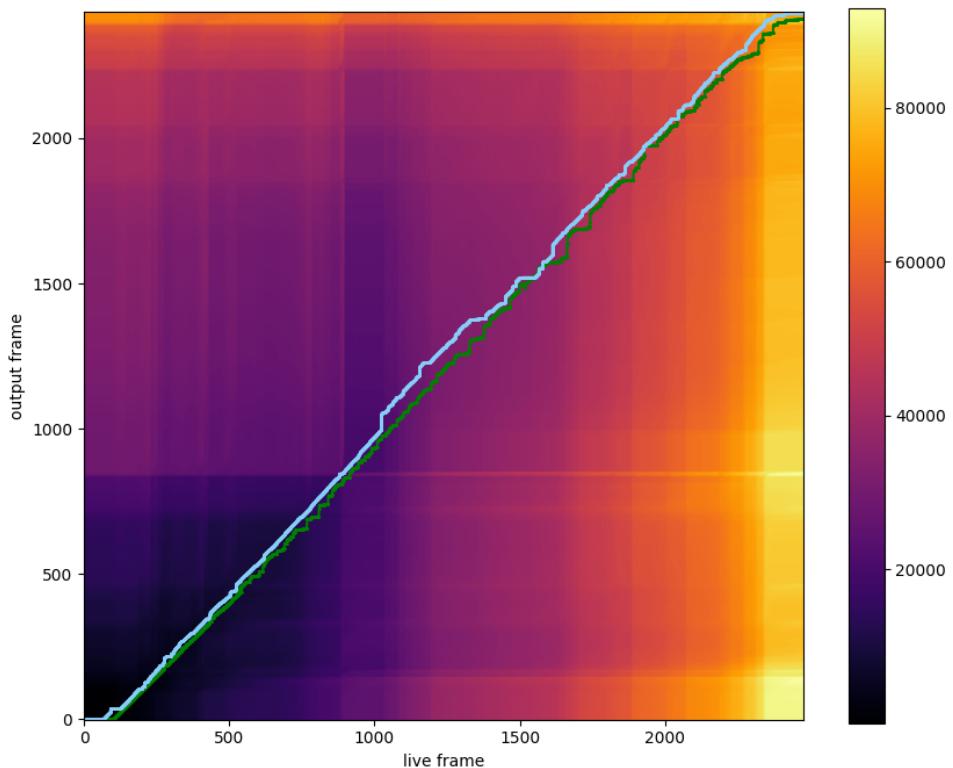


圖 3.11: DTW 與音樂追蹤模組的輸出路徑比較圖

四、 實驗設計與結果

4.1 音源分離評估

本節將比較兩個不同的音源分離模型，Aug4mss [39] 與本研究使用的 Band-Split RNN [34] 在小提琴與鋼琴混合音訊資料上的表現。首先會先介紹使用的訓練資料集與測試資料集，接著介紹模型訓練的設定、評估指標與評估結果。

4.1.1 音源分離資料集

在眾多的音訊公開資料集中，能包含多音軌錄音資料和乾淨的單音軌錄音資料可說是少之又少，在古典樂器的音訊資料集更是如此。在 [39] 中他們從網路上蒐集了 6 小時的古典小提琴獨奏錄音與 6 小時的流行鋼琴獨奏錄音作為他們的訓練與驗證資料集，但因為版權的關係，並沒有提供他們蒐集的資料集給外界使用。因此本研究也遵循 [39] 的方式，將小提琴與鋼琴兩種樂器的資料分開蒐集。我們整合了多個公開資料集作為訓練資料，公開資料集包含由 John Thickstun 等人提供的 MusicNet [56], [57]、Muneratti Ortega 等人提供的 Expressive Solo Violin [58] 與 Dong, Hao-Wen 等人提供的 Bach Violin Dataset [59]。

MusicNet 資料集中有 330 個古典音樂錄音，錄音種類包含鋼琴獨奏、小提琴獨奏、大提琴獨奏、長笛獨奏、鋼琴與樂器合奏、管樂合奏與弦樂合奏等組合，我們從資料集中挑選出鋼琴獨奏與小提琴獨奏的錄

音資料作為訓練資料集的一部分。Expressive Solo Violin 資料集是由專業的小提琴家在同一天錄製九個不同曲子片段的錄音，每一個片段會以不同的音樂表達方式演奏三次，並使用多個電容式麥克風同時錄音。我們採用了資料集中所有的錄音資料 (81 個片段) 作為訓練資料集的一部分。Bach Violin Dataset 為整合了高品質公開錄音的巴赫小提琴獨奏奏鳴曲 (BWV 1001-1006) 資料集，其中包含 17 位小提琴家在不同的演奏場所下錄製的資料，我們採用有提供錄音檔案的資料作為訓練資料集的一部分。

整合完畢的訓練資料集格式如表 4.1 所示，小提琴獨奏音檔的總時長為 5 小時 24 分 47 秒，鋼琴獨奏音檔的總時長為 5 小時 06 分 37 秒，所有資料皆為單聲道 WAVE 音訊格式。

表 4.1: 整合的訓練資料集格式

	小提琴音源	鋼琴音源
總時長	5hr 24min 47sec	5hr 6min 37sec
Channel	Mono	Mono
音訊格式	WAV	WAV

我們根據 [39] 中切割資料集的方式，將每種樂器的訓練資料集切成訓練集與驗證集，如表 4.2 所示。

表 4.2: 每種樂器的訓練集與驗證集大小

	訓練集	驗證集
小提琴	4hr	1hr 24min 47sec
鋼琴	4hr	1hr 6min 37sec

測試資料集我們採用 [39] 所提供的公開整合測試資料集進行評估，此資料集是從公開資料集 MedleyDB [60] 中特別挑選出小提琴與鋼琴的錄音所製作的測試資料集。

放資料集的連結?

4.1.2 模型訓練細節

為了公平起見，我們使用自己的公開資料集重新訓練 Aug4mss 模型來評估效能，在 Aug4mss 的論文 [39] 中透過控制每次迭帶的訓練樣本數設計了 Data-limit 與 Data-rich 兩種情境，測試模型在資料缺乏與資料足夠時的表現。因此在訓練兩種模型時，我們參考 [39] 設定的資料筆數 N ， N 為在模型訓練過程中，每次迭帶從每種樂器的資料集隨機選取 N 對所混合的資料筆數。我們針對上述兩種情況做測試。訓練與驗證的資料筆數 N 如表 4.3 所示。

表 4.3: 每次迭帶的訓練與驗證資料筆數

	Data-limit	Data-rich
訓練資料混合筆數	250 pairs	2000 pairs
驗證資料混合筆數	100 pairs	

所有訓練與測試的資料皆為 44100Hz 取樣率的單聲道音訊，STFT Window size 為 2048，Hop size 為 512，並保留原始模型的設定參數。模型訓練所使用的顯示卡為 GeForce RTX™ 3080 Ti，CPU 為 11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz。

4.1.3 音源分離評估指標

我們使用 [34] 中的兩種評估指標來評估模型，這兩種評估指標皆是基於式 (4.1) 計算訊號失真比。

$$SDR = 10 \log_{10} \frac{\sum_n \|s(n)\|^2 + \epsilon}{\sum_n \|s(n) - \hat{s}(n)\|^2 + \epsilon}, s(n) \in \mathbb{R}^2 \quad (4.1)$$

上述公式計算同一筆資料中在離散時間 n 的訊號失真比， $s(n)$ 為正確的波型， $\hat{s}(n)$ 為估計的波型，另外為了避免除數為零，我們使用一個很小的常數 $\epsilon = 10^{-7}$ 來避免這個情況。當 SDR 的值越高，代表估計的波型與正確的波型越相近，分離效果越好。

下面介紹兩種評估指標的來源與計算方法：

1. uSDR：Utterance-level SDR 為 [24] 提出的評估方法，我們使用官方的版本針對整個訊號來計算 SDR，並計算所有曲子的平均 SDR 數值。
2. cSDR：Chunk-level SDR 為 bss_eval 指標 [61] 中計算的 SDR 方法，我們使用官方的版本，計算每首曲子中所有 1 秒區塊的 SDR 中位數，獲得每首曲子的 SDR 數值後再取中位數。

4.1.4 音源分離結果比較

根據 4.1.2 節所提到的兩種情況，表 4.4 與表 4.5 為模型在 Data-limit 與 Data-rich 情況下訓練的結果。

在 [39] 中他們加入了多種 Data Augmentation 方法測試模型效能，由於本研究在訓練模型時皆採用 Random mixing 的方式從訓練資料中隨機挑選每種樂器的 10 秒音訊進行混合，因此本研究只比較 [39] 中 Random mixing 的結果。

表 4.4: Data-limit 結果數值比較

分離目標樂器	模型	cSDR	uSDR
Violin	Aug4mss(paper)	1.08	No result
	Aug4mss(retrain)	2.05	1.72
	Band-Split RNN	8.910	3.264
Piano	Aug4mss(paper)	7.43	No result
	Aug4mss(retrain)	11.906	10.974
	Band-Split RNN	14.659	13.460

在 Data-limit 的情況下，Band-Split RNN 在小提琴和鋼琴的分離效果上都是表現最為出色。Band-Split RNN 在小提琴分離上的 cSDR 和 uSDR 分別為 9.429 和 6.767，而在鋼琴分離上的 cSDR 和 uSDR 分別為 14.556 和 13.370。相比之下，Aug4mss (retrain) 的結果分別為 2.05 和 1.72 (小提琴)，以及 11.906 和 10.974 (鋼琴)。

表 4.5: Data-rich 結果數值比較

分離目標樂器	模型	cSDR	uSDR
Violin	Aug4mss(paper)	3.84	No result
	Aug4mss(retrain)	5.811	5.666
	Band-Split RNN	11.872	8.297
Piano	Aug4mss(paper)	13.46	No result
	Aug4mss(retrain)	13.514	14.124
	Band-Split RNN	16.526	15.467

在 Data-rich 的情況下，Band-Split RNN 在小提琴和鋼琴的分離效果上也是表現最為出色。Band-Split RNN 在小提琴分離上的 cSDR 和 uSDR 分別為 11.872 和 8.297，而在鋼琴分離上的 cSDR 和 uSDR 分別為 16.526 和 15.467。相比之下，Aug4mss (retrain) 的結果分別為 5.811 和 5.666 (小提琴)，以及 13.514 和 14.124 (鋼琴)。

可以注意到 Aug4mss (retrain) 的結果皆比 Aug4mss (paper) 要來的好，可能是因為我們將原本的 STFT 參數改為與 Band-Split RNN 所設定的數值相同，原本的 Window Size 從 4096 變為 2048，Hop Size 從 1024 改為 512，因此使模型能訓練到更細部的特徵。

從表 4.4 與 表 4.5 可以看出，不管是在資料缺少或是資料足夠的情況，Band-Split RNN 皆在兩種分離目標中取得最好的數值，我認為是因為 Band-Split RNN 有特別針對重要的頻帶做細部的切割，使模型能學習到更重要的特徵。

4.1.5 頻帶切割對於分離結果的影響

在 [34] 中有針對頻帶分割對模型結果的影響做一系列的測試，我們可以知道根據樂器不同的頻率範圍、音色特徵等等因素來調整頻帶分割的寬度，可能會提升模型的結果。

在 3.2.2 節我們針對不同的樂器設計不同的頻帶切割範圍，為了驗證此方法效果，我們針對自己設計的頻帶切割範圍，實作以下三個實驗，

1. 使用 [34] 中效果最好的頻帶切割範圍 V7 訓練小提琴與鋼琴的音源分離模型
2. 使用小提琴的頻帶切割範圍訓練鋼琴音源分離模型
3. 使用鋼琴的頻帶切割範圍訓練小提琴音源分離模型

為了節省訓練時間，上述實驗皆使用 Data-limit 的情況訓練，我們將這些模型的結果一一列出比較，如表 4.6 所示。

表 4.6: Data-limit 不同的 Band-Split Bandwidth 結果比較

分離目標樂器	Bandwidth	cSDR	uSDR
Violin	Violin bandwidth version	8.910	3.264
	Piano bandwidth version	8.796	3.077
	V7 bandwidth version	8.842	3.157
Piano	Violin bandwidth version	14.209	13.141
	Piano bandwidth version	14.659	13.460
	V7 bandwidth version	14.556	13.370

當目標樂器為小提琴時，我們使用小提琴的頻帶切割範圍的訓練結果優於使用鋼琴範圍與 V7 範圍，當目標樂器為鋼琴時，我們使用鋼琴的頻帶切割範圍的訓練結果優於使用小提琴範圍與 V7 範圍，由此可見，當我們根據樂器的聲音特性來分配每個頻帶的細緻度，有助於音源分離的效果。另外可以看到 V7 在兩種目標樂器的訓練結果表現皆為中間，且沒有與最好的效果差太多，因此可以說 V7 頻帶範圍是小提琴與鋼琴都適用的範圍。

4.2 音樂追蹤評估

4.2.1 系統在不同速度下的追蹤結果

我們參考 [51] 中的評估方法來檢測小提琴演奏追蹤系統的表現，在 [51] 中，現場演奏樂器為鋼琴，電腦輸出合奏為小提琴，演奏曲子為貝多芬的 Violin Sonata No.5, Op.24(Spring), movement I. Allegro 前 25 小節。為了評估系統在不同演奏速度下的反應，他們使用四種不同速度範圍測試系統，分別為演奏正常速度 (115-145bpm)、慢速 (90-120bpm)、快速 (135-175bpm)、漸快 (由 80bpm 加速至 160bpm)，參考音訊的速度設定為 120bpm。他們使用錄音設備錄製現場鋼琴與小提琴的演奏，並使用節拍追蹤工具得到現場與參考鋼琴之間的節拍位置，進一步生成正確小提琴輸出位置 (ground truth)，最後計算現場小提琴輸出位置與正確小提琴輸出位置的 DTW 對齊路徑，使用式 (4.2) 與式 (4.3) 計算瞬間延遲與平均延遲來評估。

本研究使用的現場演奏樂器為小提琴，電腦輸出合奏為鋼琴，在演奏曲子與參考音訊的設定上我們跟隨 [51] 的設定，然而在 [51] 中並沒有提到他們是如何控制現場演奏速度的變化，本研究為了能客觀地控制速度變化並計算評估結果，我們將現場演奏的部分改為使用隨機生成速度的 MIDI 檔案來評估，由於 MIDI 檔案擁有音符開始時間、節拍等等資訊，因此能精準地計算延遲，此外，我們可以透過現場 MIDI 檔案生成正確的鋼琴輸出音訊，因此不需要額外的節拍追蹤工具輔助。雖然現場演奏為預先設定好的檔案，但這並不影響即時的效果，現場音訊還是會以串流的方式進行處理。

圖 4.1為在四種不同速度範圍下隨機生成的 MIDI 檔案，我們透過讀取 MIDI 檔案中每個 MIDI 事件，將每個 MIDI 事件設定為有 50% 的機率可以在速度範圍下隨機設定速度。圖 4.1a為在慢速 (90-120bpm) 範圍的 MIDI 檔案、圖 4.1b為在正常速度 (115-145bpm) 範圍的 MIDI 檔案、

圖 4.1c 為在快速 (135-175bpm) 範圍的 MIDI 檔案，可以看到每個有設定速度的 MIDI 事件，其速度設定都在對應的範圍中。另外圖 4.1d 為漸快 (由 80bpm 加速至 160bpm) 範圍的 MIDI 檔案，我們平均的將速度範圍分配到每個 MIDI 事件上，模擬漸快的情況。

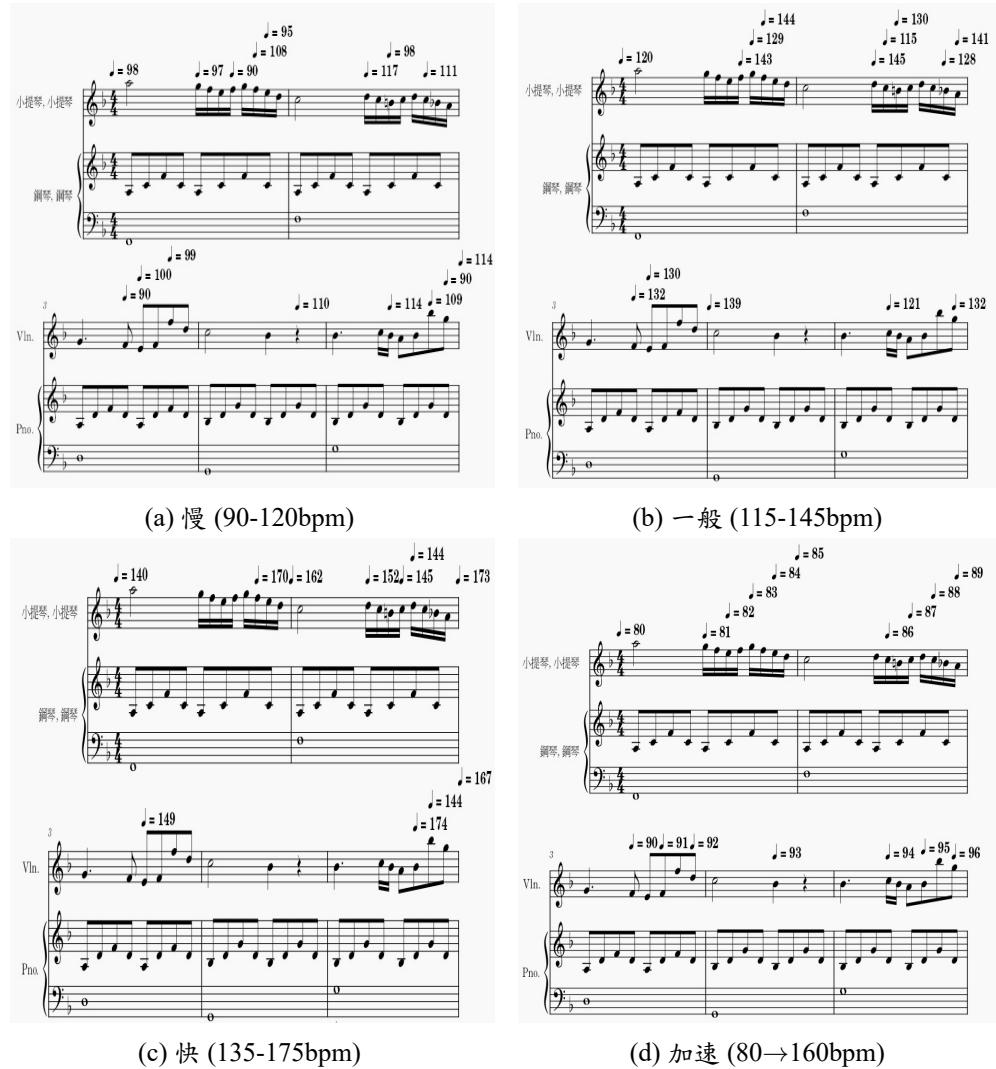


圖 4.1: 四種不同速度範圍的 MIDI 樂譜前五小節

由於正確的鋼琴輸出音訊已經透過 MIDI 檔案生成，因此我們只需要將正確伴奏音訊 Acc_{gt} 與追蹤系統估計的伴奏音訊 Acc_{out} 計算 DTW 對齊路徑，由於伴奏音訊的時間長度是一致的，因此可以根據式 (4.2) 算出 Acc_{out} 在 Acc_{gt} 上的瞬間延遲 (latency)。

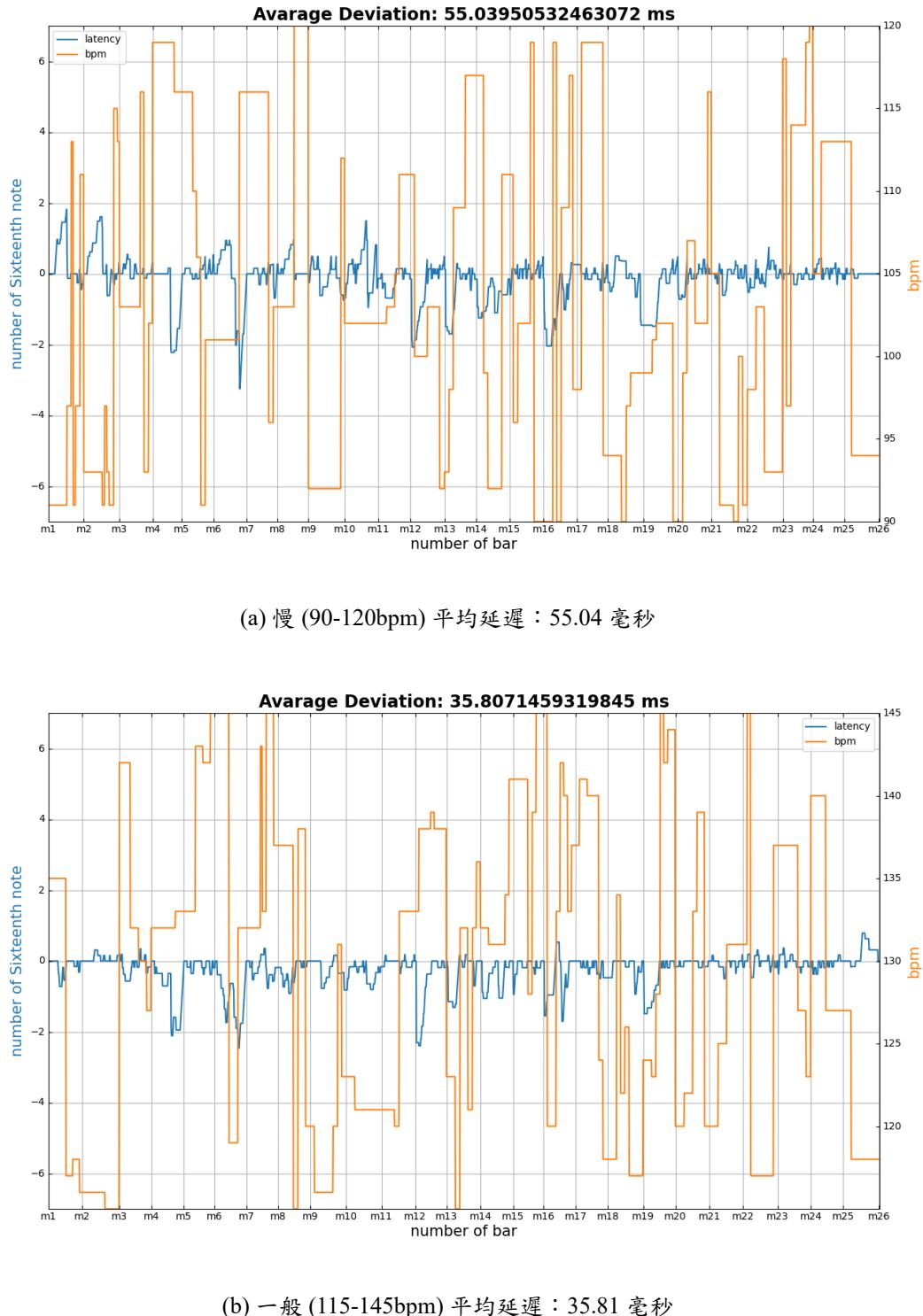
$$\Delta[t_i] := t_j - t_i \quad (4.2)$$

t_i 為 Acc_{out} 的時間點， t_j 為 Acc_{gt} 的時間點，當 $t_i = t_j$ 時，代表在這個時間點兩段音訊是對齊的，當 $t_i > t_j$ 時，代表 Acc_{out} 比 Acc_{gt} 還要快，當 $t_i < t_j$ 時，代表 Acc_{out} 比 Acc_{gt} 還要慢。

我們也用式 (4.3) 衡量整首曲子的平均延遲，計算方式為將每個時間點的瞬間延遲取絕對值後加總，並除以曲子的長度 T 。 $|\Delta t|$ 為現場伴奏音訊的延遲。

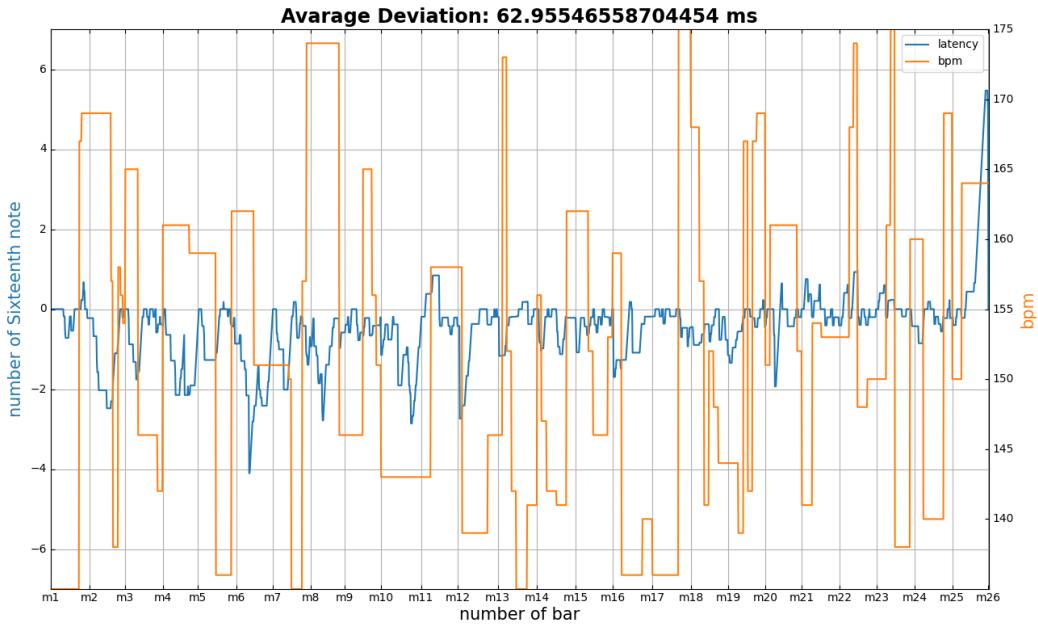
$$\frac{1}{T} \sum_{t=0}^T |\Delta[t]| \quad (4.3)$$

圖 4.2 與圖 4.3 顯示系統在四種速度變化下隨時間變化的延遲圖，也就是現場系統伴奏比正確伴奏要提前或落後多少個 16 分音符。藍色線為系統延遲的 16 分音符數量，橘色線為當下的速度 (bpm)。圖 4.2a 是慢速時系統隨時間變化的延遲，可以看到系統在一開始雖然動盪較大，但都保持在不超過 2 個 16 分音符的延遲下穩定追蹤，即使節拍動盪很大也不影響到系統的穩定性，最後的平均延遲為 55.04ms。圖 4.3a 是快速時系統隨時間變化的延遲，與慢速的延遲圖很像，但動盪較大，這可能是因為快速的速度範圍與參考的速度相差較多，因此導致系統在追蹤時較容易有延遲，最後的平均延遲為 62.96ms。圖 4.3b 是漸快時系統隨時間變化的延遲，由於一開始速度較慢，因此導致系統剛開始追蹤時動盪較大，當速度慢慢回升到接近參考音訊的速度時，可以看到動盪越來越小，最後的平均延遲為 58.23ms。另外可以發現在第五小節 (m5) 與第七小節 (m7) 時，四種速度的延遲皆為正延遲，也就是比起正確的時間點還要提前，可能是因為剛好那個時間點為休止符，也就是沒有音樂，因此造成系統先對齊到了最後面的靜音時間點並等待，導致延遲。

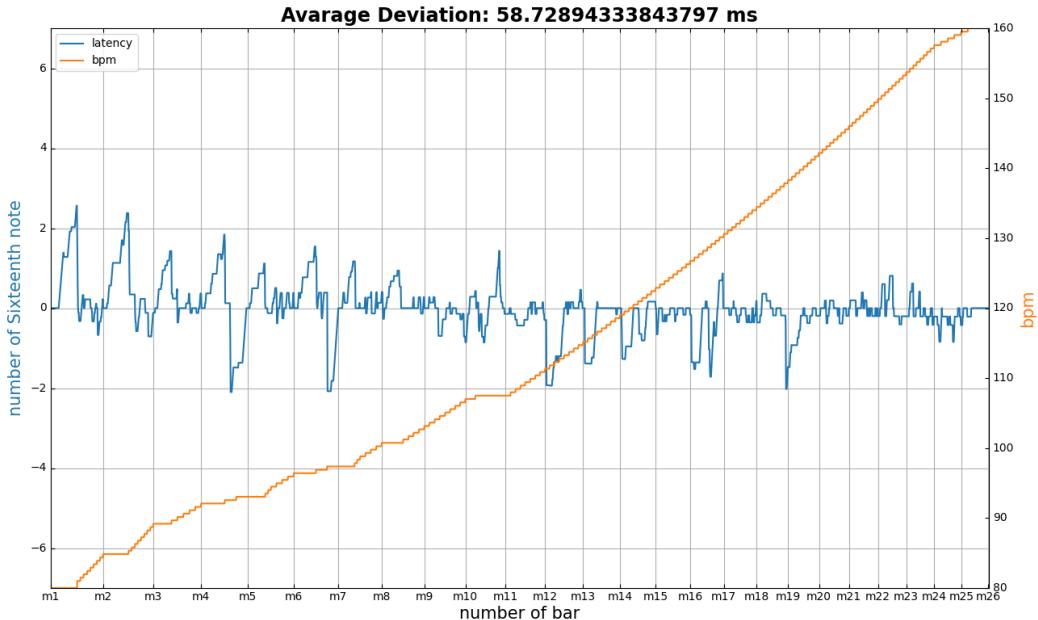


(b) 一般 (115-145bpm) 平均延遲：35.81 毫秒

圖 4.2: 系統在慢、一般速度的延遲時間



(a) 快 (135-175bpm) 平均延遲：62.96 毫秒



(b) 加速 (80→160bpm) 平均延遲：58.73 毫秒

圖 4.3: 系統在快、加速速度的延遲時間

另外我們也顯示了四種速度的對齊路徑結果圖，橫軸為現場音訊的時間點，縱軸為參考音訊的時間點，熱點圖的顏色代表在時間點 (i, j) 的累積距離成本值，成本值越高顏色越接近黃色。(a) 圖為追蹤系統在執行

時所輸出的結果圖，(b) 圖為離線對齊路徑與線上對齊路徑的比較圖，離線對齊路徑使用 DTW 計算。

圖 4.4 在大部分的時間點，速度會比參考音訊的速度要慢，因此可以看到圖 4.4a 在某些區間（例如 live 約 600frames 時）綠色的點所連成的線較為平緩，代表此區間系統輸出較多相同的參考音訊時間點來等待現場音訊。在圖 4.4b 的表現上，幾乎與離線對齊時的最佳路徑是一致的。

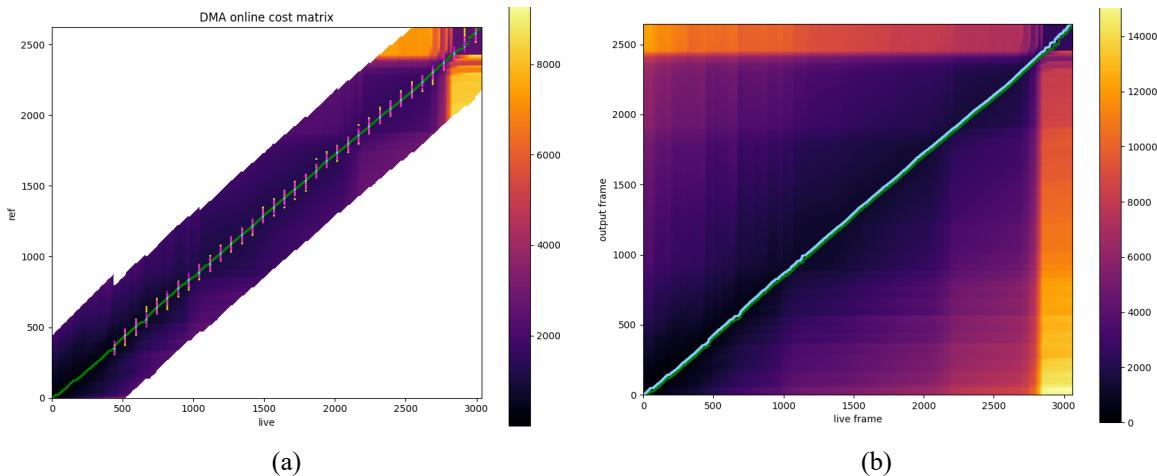


圖 4.4: 慢 (90-120bpm)

圖 4.5的速度與參考音訊是最接近的，因此在圖 4.5a幾乎不會有波動，圖 4.5b 也是表現得與離線對齊路徑相當。

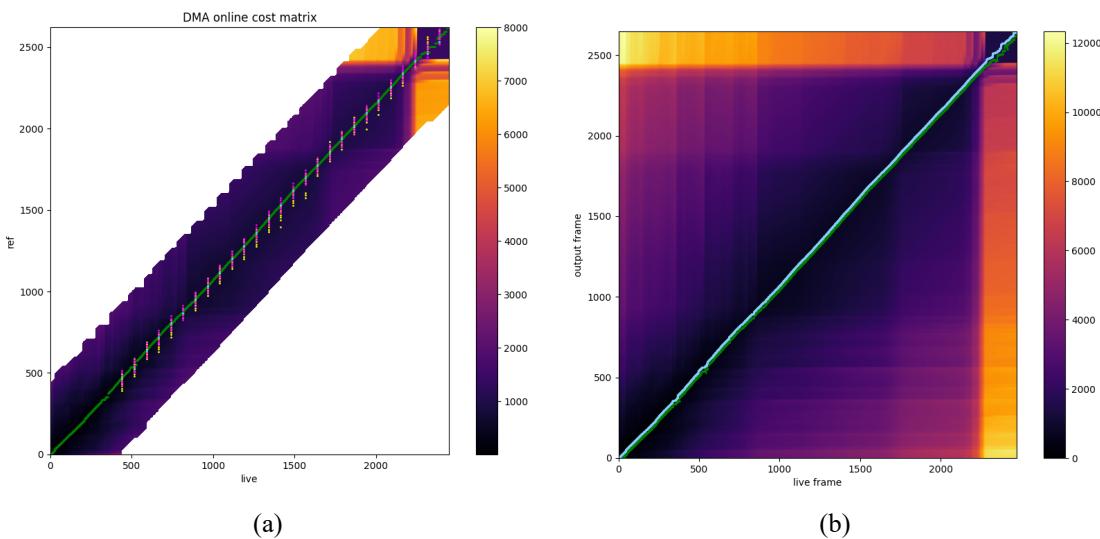


圖 4.5: 一般 (115-145bpm)

圖 4.6在大部分的時間點，速度會比參考音訊的速度要快，因此可以

看到圖 4.6a在某些區間 (例如 live 約 450frames 時) 輸出路徑會呈現斷斷續續的點，代表此區間系統在輸出時認為目前現場音訊的時間點必須對齊到更後面的參考音訊，在圖 4.6b的表現上，雖然參考音訊的時間點並不是連續的，但可以看到整體路徑與離線對齊路徑還是一致的。

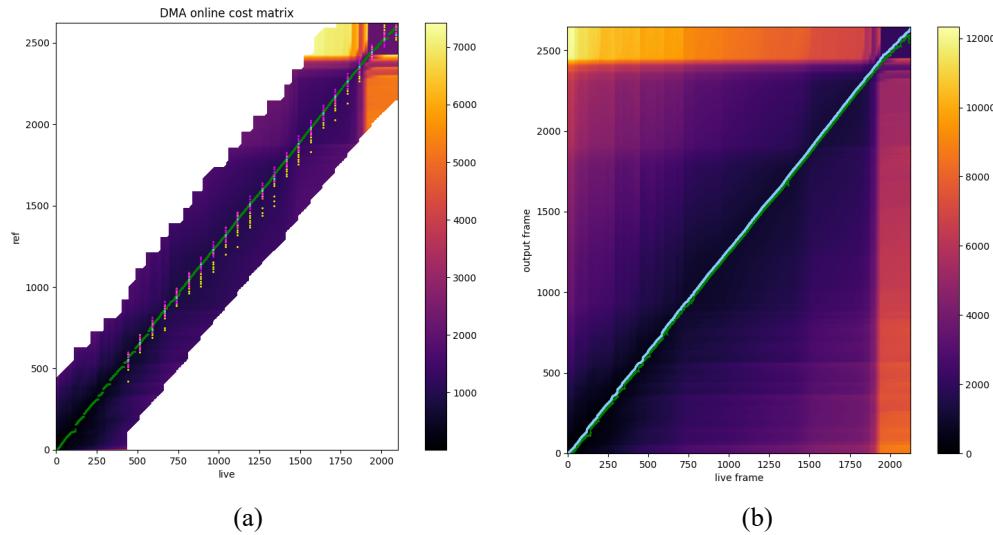


圖 4.6: 快 (135-175bpm)

圖 4.7涵蓋了整個慢到快的速度區間，可以看到圖 4.7a在前面速度較慢時輸出相同的點的狀況較多，當現場音訊演奏越來越快，輸出點所連成的線的斜率越來越高。在圖 4.7b的表現上，與離線對齊路徑也是呈現一致。

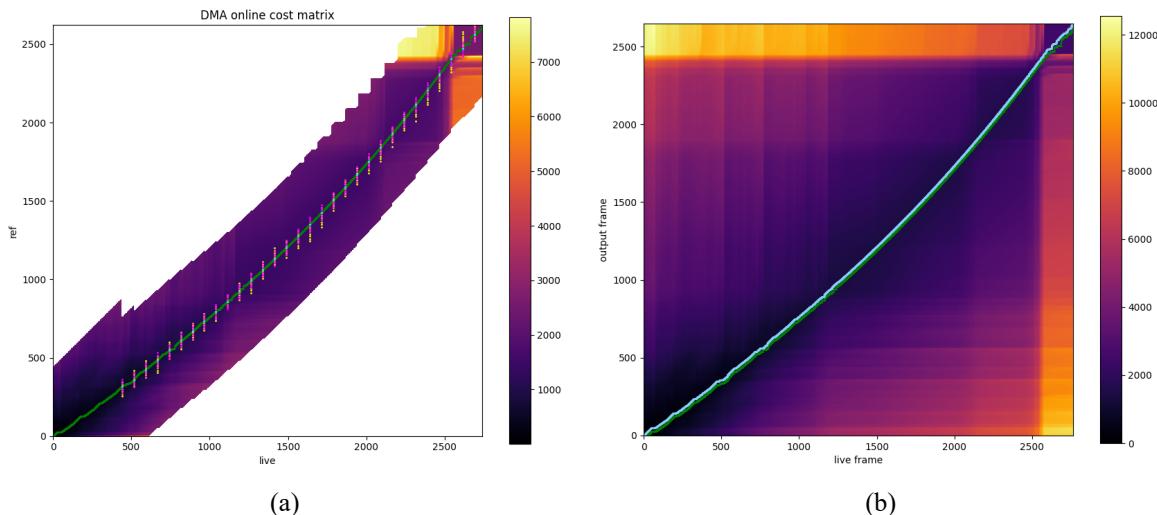


圖 4.7: 加速 (80→160bpm)

上述結果的音訊檔案可從 [github](#) 聆聽。

4.2.2 系統在使用音源分離音訊做為參考特徵的追蹤結果

為了評估系統是否在不同特徵下也能有好的追蹤結果，我們使用與4.2.1節相同的曲子，在網路上找到由 kopikostar 演奏的小提琴與鋼琴合奏版本 [62]，並對此混合音訊做音源分離，將分離結果作為參考音訊與伴奏音訊。分離結果的音檔可至此連結聆聽 [github](#)。

我們實際演奏現場音訊，並演奏了四種不同速度，分別為 110bpm、120bpm、130bpm 與 140bpm，此四種速度為這首曲子最常出現的演奏速度。演奏時，為了保持演奏速度的平穩，我們皆使用節拍器來輔助，我們也演奏了自由速度版本，也就是不使用節拍器，與鋼琴互相配合演奏。在演奏時使用 AKG C519 M 電容式麥克風收錄音訊，監聽耳機為 Sony MDR 7506，並使用 Focusrite Scarlett 2i2 錄音介面處理類比與數位訊號。

由於沒有正確的伴奏音訊 (Ground truth)，因此我們無法使用4.2.1節的方法來評估效果。我們改由計算系統輸出路徑與 DTW 離線對齊路徑的平均延遲幀數來評估系統在線上追蹤的表現，這邊的平均延遲幀數代表的是對於線上追蹤表現，是否提前或落後離線追蹤表現。

下圖為五種不同速度的對齊路徑結果圖，橫軸為現場音訊的時間點，縱軸為參考音訊的時間點，(a) 圖為追蹤系統在執行時所輸出的結果圖，(b) 圖為離線對齊路徑與線上對齊路徑的比較圖與平均延遲幀數。

圖 4.8為速度 110bpm 下的追蹤結果，由圖 4.8a的 xy 軸可以看出現場音訊的速度比參考音訊要來的慢，因此會出現等待的情況，例如 live frame=700 的時候。圖 4.8b顯示了追蹤結果與離線計算結果的延遲為 26.62 幀，約為 532ms 左右。

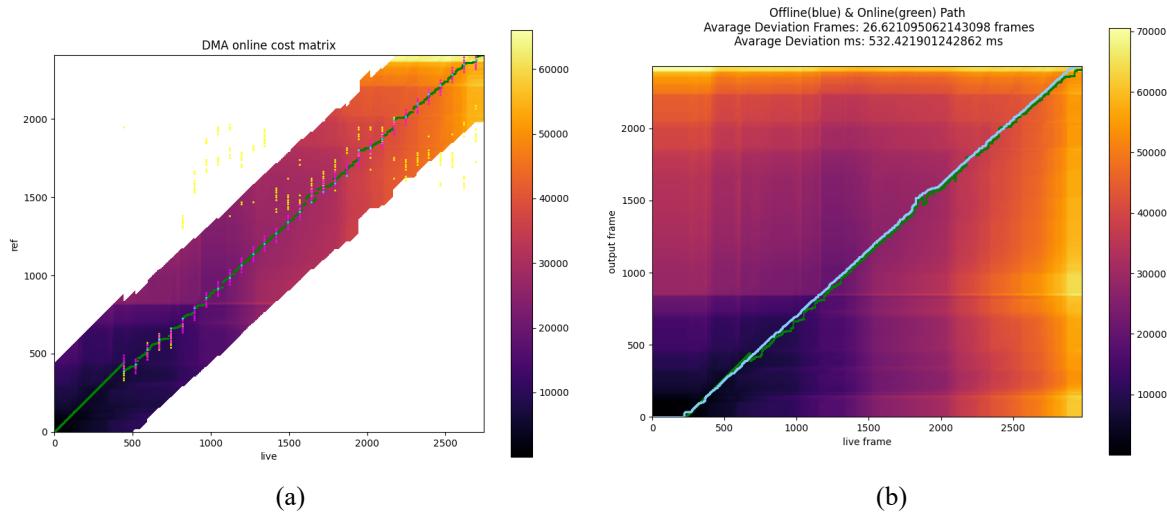


圖 4.8: 110bpm 平均延遲：26.62 帖

圖 4.9為速度 120bpm 下的追蹤結果，由圖 4.9a的 xy 軸可以看出現場音訊的速度也是比參考音訊的速度稍慢，圖 4.9b顯示了追蹤結果與離線計算結果的延遲為 27.98 帖，約為 560ms 左右。

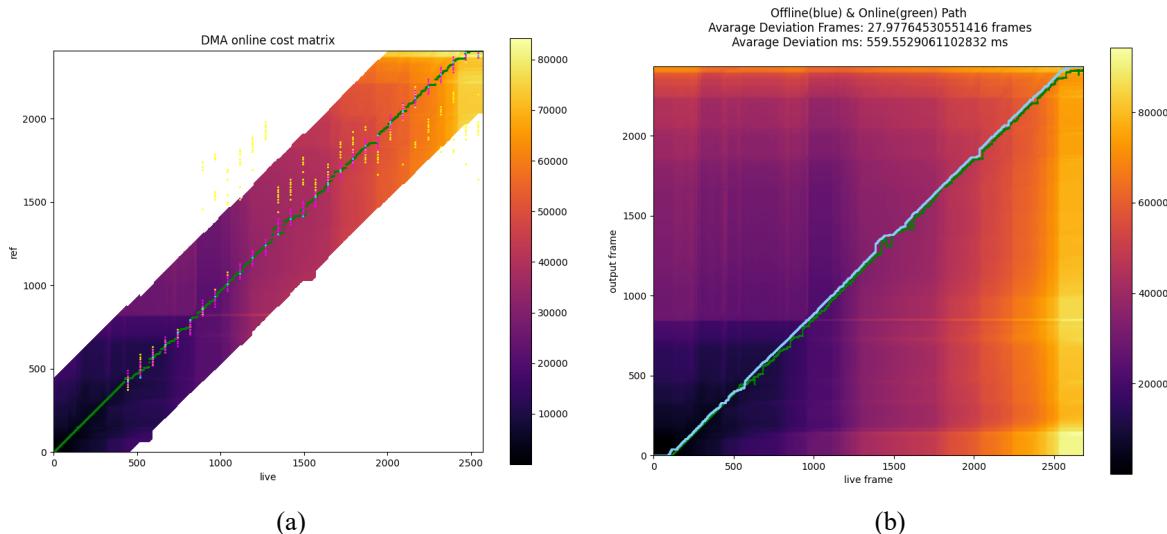


圖 4.9: 120bpm 平均延遲：27.98 帖

圖 4.10為速度 130bpm 下的追蹤結果，由圖 4.10a的 xy 軸可以看出現場音訊與參考音訊的速度較為相近，但可能因為特徵差距的關係使追蹤結果並沒有比起前面的結果要好。圖 4.10b顯示了追蹤結果與離線計算結果的延遲為 37.49 帖，約為 750ms 左右。

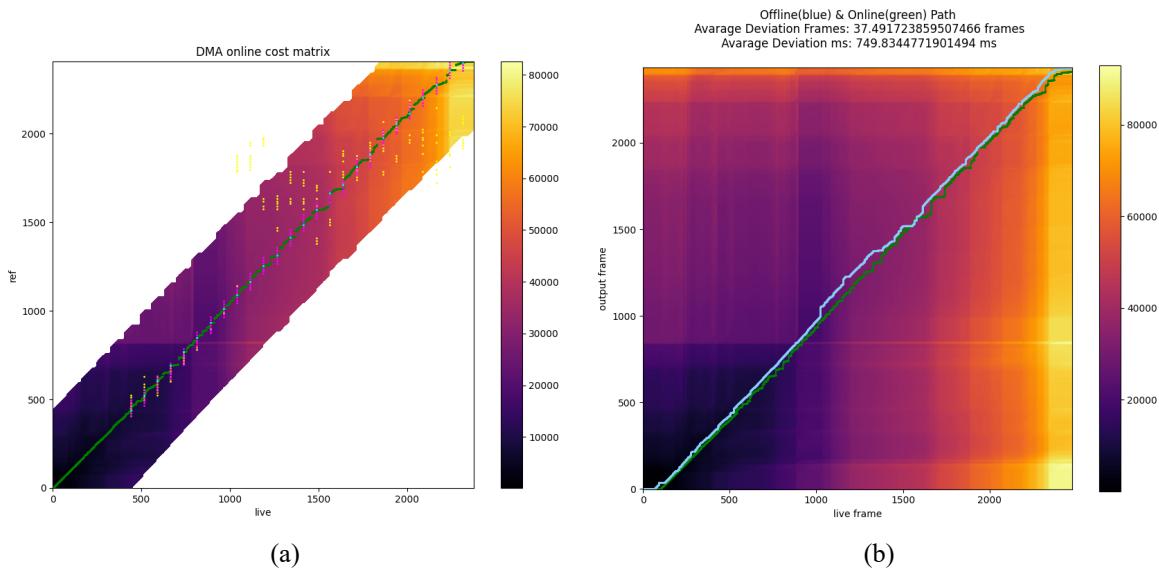


圖 4.10: 130bpm 平均延遲：37.49 帖

圖 4.11為速度 140bpm 下的追蹤結果，由圖 4.11a的 xy 軸可以看出現場音訊比參考音訊的速度稍快一些，在 live frame 約 500 的時候，可以看到追蹤的路徑並不穩定，此時靠著 RPE 的粗略估計位置將 ODTW 對齊錯誤的地方拉回正軌。圖 4.11b顯示了追蹤結果與離線計算結果的延遲為 40.68 帖，約為 814ms 左右。

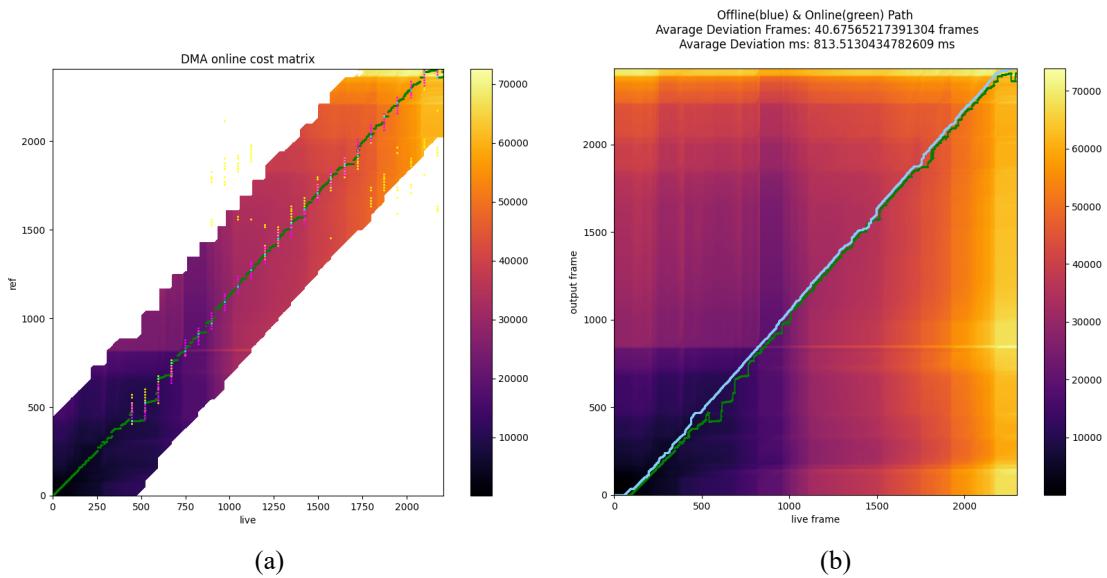


圖 4.11: 140bpm 平均延遲：40.68 帖

圖 4.12為自由速度下的追蹤結果，由圖 4.12a的 xy 軸可以看出現場音訊比參考音訊的速度稍慢一些，在 live frame 約 1500 之後，可以看到

追蹤的路徑並不穩定，我認為可能是這段的旋律是重複的，因此可能在粗略估計的位置上無法精準的計算目前粗估位置，但最後還是有走回正軌。圖 4.12b 顯示了追蹤結果與離線計算結果的延遲為 28.43 帖，約為 569ms 左右。

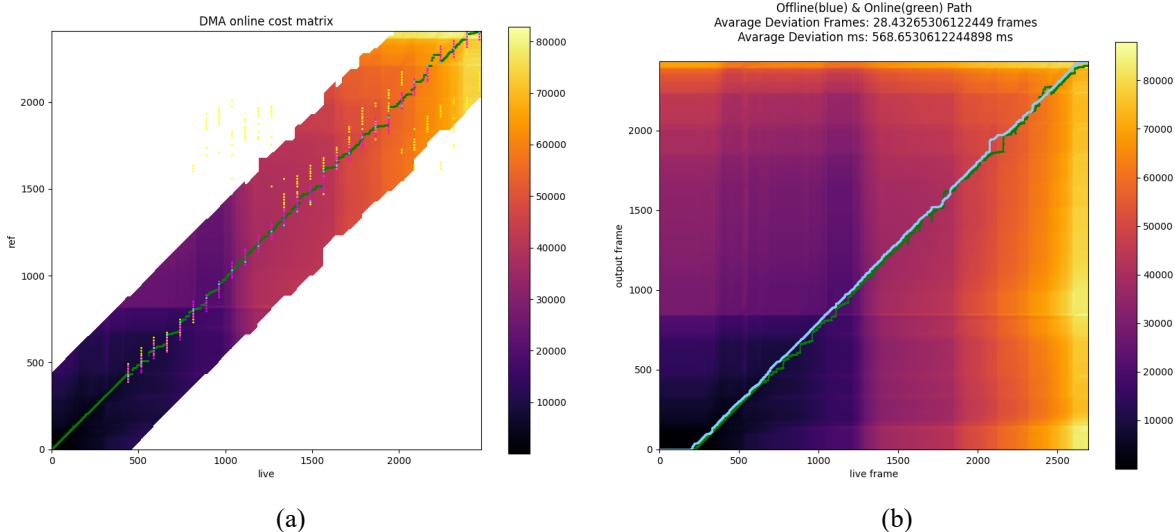


圖 4.12: 自由速度平均延遲：28.43 帖

由於參考音訊與現場音訊的特徵不一致，導致 ODTW 在計算對齊點時穩定性較為不足，RPE 估計出的位置也因為兩個音訊的特徵不夠相近，使粗略估計的位置偏差較嚴重，因此追蹤結果圖大多數看起來比 4.2.1 節的結果要差。但可以看到在比較離線對齊路徑與線上對齊路徑的結果圖，大致上的路徑都與離線對齊路徑重合，因此我們可以知道這兩個音訊在線上的最佳對齊路徑已經非常接近離線對齊路徑，達到了即使使用不同特徵追蹤也能維持離線最佳對齊的效果。另外可以看到比較圖中從第 0 帖到兩段路徑開始對齊的時間點是一致的，代表我們的 Music detector 在判斷演奏者的演奏時間是精準的。此外，我認為平均延遲都高達 500ms 以上可能是因為 DTW 在計算對齊路徑時， i 與 j 都有遵守連續性的限制，因此不會出現一次跳 2 格以上的路徑，且 i 是可以重複的；但我們在計算 ODTW 時，在 j 方向是有可能出現一次跳 2 格以上的路徑，然而 i 為嚴格遞增，因此造成兩個路徑在計算時，因為固定了 i 的位置，在 $(t_j - t_i)$ 的差距可能會更大，但實際聽起來並沒有延遲 500ms 那麼多。

我們也測試了使用同一人演奏的參考特徵來追蹤現場演奏的效果，首先我們先錄製了與伴奏音訊完整對齊的音訊作為參考音訊，接著現場演奏我們一樣採自由速度，聆聽伴奏音訊來演奏，結果如圖 4.13 所示，可以看到線上對齊路徑與離線對齊路徑重合，代表系統輸出的路徑是最佳對齊位置，在平均延遲上也比圖 4.12 要來的好。

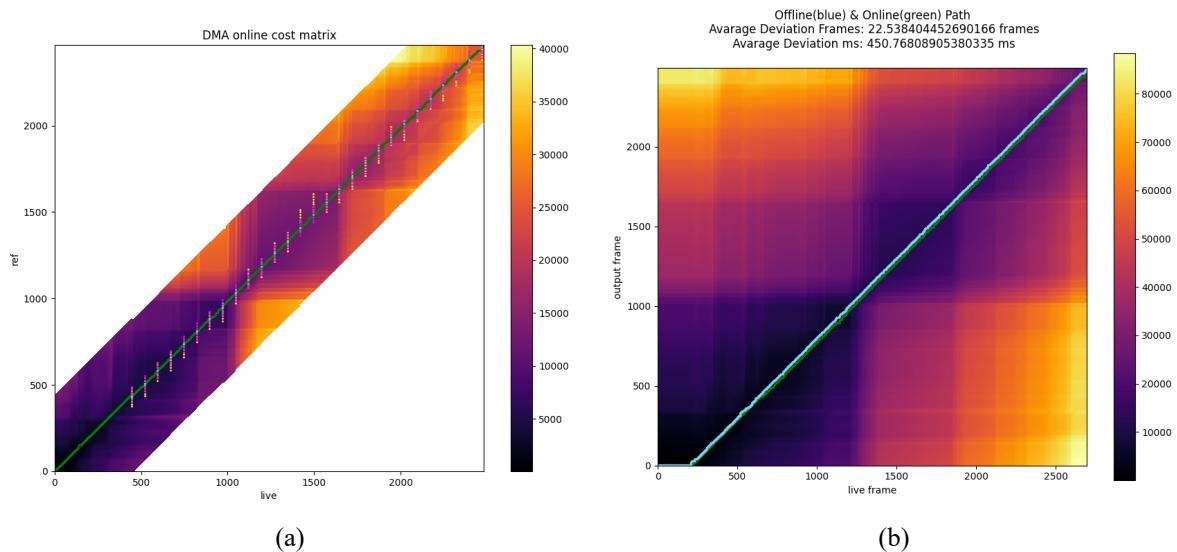


圖 4.13: 自由速度 (參考音訊為同一人所演奏) 平均延遲：22.54 帖

五、總結

5.1 結論

本研究提出了一套結合音源分離模組與音樂追蹤模組的小提琴演奏追蹤系統，此系統能將小提琴與鋼琴的混合音源分離成參考音源與伴奏音源，並結合現場小提琴音訊實現即時追蹤與伴奏的效果。在音源分離模組方面，本研究採用 Band-Split RNN 模型並提出了簡單的頻帶切割方法，有效提升模型的分離效果。在音樂追蹤模組方面，本研究改進了音訊特徵擷取方式，即時處理串流音訊使其他模塊能使用音訊特徵。此外，我們也改良了音樂偵測模塊，加入了平均振幅閥值判斷法加速過濾靜音片段。我們也改進了線上動態時間規整演算法 (ODTW)，使系統能夠更精確地對齊現場音訊並即時輸出。最後，改進了貪心向後對齊演算法 (GBA)，使模組在決定輸出位置時能參考更精準的資訊。

研究實驗結果顯示，音源分離模型在資料缺乏與資料充足的情況下，效果皆優於現有的基線模型。針對加入頻帶切割方法是否對模型有正面影響的實驗結果表明，對不同樂器使用頻帶切割方法可提升模型效果。另外，音樂追蹤模組的實驗結果顯示，在隨機速度下的即時追蹤效果最好的延遲為 35.81ms，最差的結果也只延遲了 4 個 16 分音符，即使從慢版速度 (80bpm) 加速到快板速度 (160bpm)，即時追蹤效果也同樣穩定。在不同特徵的即時追蹤效果，雖然比相同特徵的效果要差，但與離線追蹤效果相比，即時追蹤的最佳路徑幾乎與離線追蹤的最佳路徑重疊，顯示即時追蹤效果也能達到與離線追蹤相同的穩定度與精準度。

5.2 未來展望

本研究成功在使用不同特徵且即時的情況下追蹤小提琴演奏的位置並輸出伴奏音訊，但此系統還有許多可以改進的地方，未來的研究可以探索以下幾個方向。

在音源分離方面，本研究使用了較輕量的 Band-Split RNN 模型進行訓練，未來可以嘗試使用更大的模型，例如目前效果最佳的模型 Hybrid Transformers [37]。此模型能同時在時頻域和波形域進行訓練，且不需要訓練多個分離目標模型。此外，本研究只針對小提琴與鋼琴的混合音源進行研究，但其實在古典音樂中，還有許多不同的樂器合奏配置。因此，未來希望能擴展音源分離技術以涵蓋更多種類的古典樂器。

在音樂追蹤方面，由於輸出的位置可能會連續停留在同一位置，導致伴奏聽起來有失真或雜訊的感覺，未來可以研究輸出音訊的平滑化技術，或是從輸出位置估計目前的演奏速度來調整整段伴奏的快慢。另外，可以結合數位音訊工作站 (DAW)，將輸入與輸出皆透過 DAW 處理，使程式更簡化，並且能更快速地轉換特徵，從而降低系統延遲。

希望未來能在這些方面進行深入研究，以進一步提升系統的性能和應用範圍。

參考文獻

- [1] C.A.P.E. “Statistics on the number of applicants for music subjects over the years.” (2024), [Online]. Available: <https://www.cape.edu.tw/statistics/> (visited on 05/07/2024).
- [2] 我是江老師.“鋼琴伴奏月薪？一小時賺多少？到底都在做什麼？”(2020), [Online]. Available: <https://youtu.be/8MBaTBXLzEw?t=100> (visited on 05/07/2024).
- [3] A. Défossez, N. Usunier, L. Bottou, and F. Bach, *Music source separation in the waveform domain*, 2021.
- [4] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F.-R. Stöter, “Musical source separation: An introduction,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2019.
- [5] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimalakis, D. FitzGerald, and B. Pardo, “An overview of lead and accompaniment separation in music,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.
- [6] M. E. P. Davies, “Towards automatic rhythmic accompaniment,” Ph.D. dissertation, Citeseer, 2007.
- [7] Y. Li, “Application of computer-based auto accompaniment in music education,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 6, pp. 140–151, 2020.
- [8] X. Zhang and C. Liu, “Design of piano automatic accompaniment system based on artificial intelligence algorithm,” in *International Conference on Computational Finance and Business Analytics*, Springer, 2023, pp. 249–258.
- [9] N. Orio, S. Lemouton, and D. Schwarz, “Score following: State of the art and new developments,” *New Interfaces for Musical Expression (NIME)*, 2003.
- [10] M. Dorfer, A. Arzt, and G. Widmer, “Towards score following in sheet music images,” *arXiv preprint arXiv:1612.05050*, 2016.
- [11] S. Ji, J. Luo, and X. Yang, “A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions,” *arXiv preprint arXiv:2011.0680* 2020.

- [12] C. Hernandez-Olivan and J. R. Beltran, “Music composition with deep learning: A review,” *Advances in speech and music technology: computational aspects and applications*, pp. 25–50, 2022.
- [13] A. Solanki and S. Pandey, “Music instrument recognition using deep convolutional neural networks,” *International Journal of Information Technology*, vol. 14, no. 3, pp. 1659–1668, 2022.
- [14] K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar, and P. Harish, “Predominant musical instrument classification based on spectral features,” in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 2020, pp. 617–622.
- [15] E. Manilow, G. Wichern, and J. Le Roux, “Hierarchical musical instrument separation.,” in *ISMIR*, 2020, pp. 376–383.
- [16] P. Mangla, “Spotify music recommendation systems,” in *PyImageSearch*, P. Chugh, A. R. Gosthipaty, S. Huot, K. Kidriavsteva, and R. Raha, Eds., 2023.
- [17] Ableton. “Ableton live 11 lite.” (2024), [Online]. Available: <https://www.ableton.com/en/live/> (visited on 06/04/2024).
- [18] Apple. “Logic pro.” (2024), [Online]. Available: <https://www.apple.com/tw/logic-pro/> (visited on 06/04/2024).
- [19] PreSonus. “Studio one.” (2024), [Online]. Available: <https://www.presonus.com/en/studio-one.html> (visited on 06/04/2024).
- [20] Ronimusic. “Amazing slow downer.” (2024), [Online]. Available: <https://www.ronimusic.com/> (visited on 06/04/2024).
- [21] FORSCORE. “Forscore turbocharge tour sheet music.” (2024), [Online]. Available: <https://forscore.co/> (visited on 06/04/2024).
- [22] ISMIR. “International society for music information retrieval.” (2024), [Online]. Available: <https://ismir.net/> (visited on 05/06/2024).
- [23] X. Zhao, Q. Tuo, R. Guo, and T. Kong, “Research on music signal processing based on a blind source separation algorithm,” *Annals of Emerging Technologies in Computing (AETiC)*, vol. 6, no. 4, 2022.
- [24] Y. Mitsufuji, G. Fabbro, S. Uhlich, and F.-R. Stöter, *Music Demixing Challenge 2021*, 2021.
- [25] G. Fabbro, S. Uhlich, C.-H. Lai, *et al.*, “The Sound Demixing Challenge 2023 Music Demixing Track,” *arXiv e-prints*, arXiv:2308.06979, arXiv:2308.06979, Aug. 2023.
- [26] Z. Wang, K. Zhang, Y. Wang, *et al.*, “Songdriver: Real-time music accompaniment generation without logical latency nor exposure bias,” in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 1057–1067.
- [27] F. Ding and Y. Cui, “Museflow: Music accompaniment generation based on flow,” *Applied Intelligence*, vol. 53, no. 20, pp. 23 029–23 038, 2023.

- [28] C. Brazier and G. Widmer, “Improving real-time score following in opera by combining music with lyrics tracking,” *arXiv preprint arXiv:2110.02592*, 2021.
- [29] Antescofo. “Metronautapp.” (2024), [Online]. Available: <https://metronautapp.com/zh-TW> (visited on 06/04/2024).
- [30] P. Comon, “Independent component analysis, a new concept?” *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.
- [31] D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” *Advances in neural information processing systems*, vol. 13, 2000.
- [32] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, “Open-unmix - a reference implementation for music source separation,” *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019.
- [33] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimalakis, and R. Bittner, *The MUSDB18 corpus for music separation*, Dec. 2017.
- [34] Y. Luo and J. Yu, “Music Source Separation with Band-split RNN,” *arXiv e-prints*, arXiv:2209.15174, arXiv:2209.15174, Sep. 2022.
- [35] D. Stoller, S. Ewert, and S. Dixon, “Wave-u-net: A multi-scale neural network for end-to-end audio source separation,” *arXiv preprint arXiv:1806.03185*, 2018.
- [36] A. Défossez, “Hybrid spectrogram and waveform source separation,” *arXiv preprint arXiv:2111.03600*, 2021.
- [37] S. Rouard, F. Massa, and A. Défossez, “Hybrid transformers for music source separation,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.
- [38] M. Miron, J. Janer Mestres, and E. Gómez Gutiérrez, “Generating data to train convolutional neural networks for classical music source separation,” in *Lokki T, Pätynen J, Välimäki V, editors. Proceedings of the 14th Sound and Music Computing Conference; 2017 Jul 5-8; Espoo, Finland. Aalto: Aalto University; 2017. p. 227-33.*, Aalto University, 2017.
- [39] C.-Y. Chiu, W.-Y. Hsiao, Y.-C. Yeh, Y.-H. Yang, and A. Wen-Yu Su, “Mixing-Specific Data Augmentation Techniques for Improved Blind Violin/Piano Source Separation,” *arXiv e-prints*, arXiv:2008.02480, arXiv:2008.02480, Aug. 2020.
- [40] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, “Spleeter: A fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.
- [41] M. Heydari and Z. Duan, “Don’t look back: An online beat tracking method using rnn and enhanced particle filtering,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 236–240.

- [42] M. Goto and Y. Muraoka, “Musical understanding at the beat level: Real-time beat tracking for audio signals,” in *Computational auditory scene analysis*, CRC Press, 2021, pp. 157–176.
- [43] B. Di Giorgi, M. Mauch, and M. Levy, “Downbeat tracking with tempo-invariant convolutional neural networks,” *arXiv preprint arXiv:2102.02282*, 2021.
- [44] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, “Score following as a multi-modal reinforcement learning problem.,” *Trans. Int. Soc. Music. Inf. Retr.*, vol. 2, no. 1, pp. 67–81, 2019.
- [45] P. Cano, A. Loscos, and J. Bonada, “Score-performance matching using hmms,” in *ICMC*, Citeseer, 1999.
- [46] A. Arzt, G. Widmer, and S. Dixon, “Adaptive distance normalization for real-time music tracking,” in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 2689–2693.
- [47] C. Raffel and D. P. W. Ellis, “Optimizing dtw-based audio-to-midi alignment and matching,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 81–85.
- [48] I.-C. Wei and L. Su, “Online music performance tracking using parallel dynamic time warping,” in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, 2018, pp. 1–6.
- [49] S. Dixon, “Live tracking of musical performances using on-line time warping,” in *Proceedings of the 8th International Conference on Digital Audio Effects*, Citeseer, vol. 92, 2005, p. 97.
- [50] A. Arzt and G. Widmer, “Towards effective ‘any-time’ music tracking,” in *Proceedings of the 2010 Conference on STAIRS 2010: Proceedings of the Fifth Starting AI Researchers’ Symposium*, NLD: IOS Press, 2010, pp. 24–36.
- [51] Y.-J. Lin, H.-K. Kao, Y.-C. Tseng, M. Tsai, and L. Su, “A human-computer duet system for music performance,” in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM ’ 20, ACM, Oct. 2020.
- [52] Python. “Multiprocessing —process-based parallelism.” (2024), [Online]. Available: <https://docs.python.org/3/library/multiprocessing.html> (visited on 06/02/2024).
- [53] Python. “Pyaudio package.” (2024), [Online]. Available: <https://people.csail.mit.edu/hubert/pyaudio/> (visited on 06/02/2024).
- [54] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.
- [55] B. McFee, M. McVicar, D. Faronbi, *et al.*, *Librosa/librosa: 0.10.2.post1*, 2024.

- [56] J. Thickstun, Z. Harchaoui, and S. M. Kakade, “Learning features of music from scratch,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [57] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, “Invariances and data augmentation for supervised music transcription,” in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.
- [58] F. J. Muneratti Ortega, *Expressive solo violin*, 2021.
- [59] H.-W. Dong, C. Zhou, T. Berg-Kirkpatrick, and J. McAuley, *Bach violin dataset*, 2021.
- [60] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, “Medleydb: A multitrack dataset for annotation-intensive mir research,” Oct. 2014.
- [61] E. Vincent, R. Gribonval, and C. Févotte, “Performance measurement in blind audio source separation,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, pp. 1462–1469, Aug. 2006.
- [62] kopikostar. “Beethoven’s ”spring sonata” op.24 allegro.” (2024), [Online]. Available: <https://youtu.be/uDSfijK1qxo?list=PLc0i4xi7nsQRG0UTdRKrc2bxjh9pHYYJw> (visited on 06/03/2024).