ONLINE MUSIC PERFORMANCE TRACKING USING PARALLEL DYNAMIC TIME WARPING

I-Chieh Wei and Li Su

Institute of Information Science, Academia Sinica, Taiwan

ABSTRACT

Resource allocation is a critical issue in the implementation of a portable, low-latency and efficient system for interactive experience. To address this, we propose the parallel dynamic time warping (PDTW), that utilizes multi-thread computation on a modern multi-core system, for online audio-to-audio music alignment. We also discuss the evaluation methodology that benchmarks the trade-offs among latency, accuracy of alignment, and computing resource. The proposed system utilizes multiple DTW alignment processes and parallel computing technique to reduce the processing time, as well as to improve both the alignment accuracy and robustness to tempo variation. Evaluation on several datasets with artificial time stretching exhibits the capacity of the system in enriched concert experience.

Index Terms— music information retrieval, online audio alignment, parallel computing.

1. INTRODUCTION

In the era of experience economy, multimedia systems for real-time interaction are becoming ubiquitous in our lives. Music is one of the most fundamental elements that enriches such possibilities. For example, rhythm games [1, 2], page turners for music practice [3], automatic accompaniment systems [4], and enriched music experience [5, 6] have gained huge popularity, and also marks the role of the end-device-based interactive systems in the "Internet of musical things" (IoMT) [7].

It is therefore required an online alignment algorithm that can synchronize a live music performance to a target sequence, such as a reference score, accompaniment, and others, with unnoticeable latency, light computational resource, high accuracy, and preferably, easy-to-implement architecture [4–6, 8–16].[1] Despite its development for decades, building a stable music alignment system for real-time tracking of live performance is still challenging for many reasons: First, the system should reliably predict future score events with unnoticeable latency. This is hard because a music piece can be played with various speeds, loudness, and expression by different musicians and under different environments. Such a fact usually results in a high variation of *instantaneous tempo* estimation in the expressive parts of a music piece among different versions of performance in a music piece. According to an illustration in [13], an abrupt slow-down or speed-up by twice in live music performance is not uncommon. It is therefore required a stable algorithm that can track slow-varying tempo, but can still react to abrupt tempo change.

Second, hardware issues such as the I/O latency and environmental noise could be even uncontrollable when moving the system into the wild, making it difficult to assess the quality of an online alignment system. Up to date, a systematic evaluation that includes the performance of latency [8] and even computational loading for an online alignment algorithm are relatively less emphasized in previous work.

To address the above-mentioned issues, we propose the parallel dynamic time warping (PDTW) method, which performs multiple dynamic time warping (DTW) subroutines in parallel on overlapped audio segments, to estimate stable and accurate instantaneous tempo. Since multi-core CPUs are becoming ubiquitous in the laptops and mobile devices, we emphasize that such an approach is practical in developing interactive music systems. For an in-depth study, we evaluate the trade-offs among accuracy, latency, and computation loading on a synthetic dataset with approach similar to [17].

2. BACKGROUND AND RELATED WORK

Most of the online alignment systems infer the score position by tracking the instantaneous tempo or detecting the onsets of important events [13]. In general, the performance of these algorithms are constrained by the following three factors:

- The *intrinsic latency* caused by a limited length of a segment for instantaneous tempo estimation [18]. For example, if we use a 8-sec segment to estimate the tempo, this tempo is actually not an "instantaneous tempo," but an averaged one over the previous 8 seconds, and is (on average) by 4 seconds delayed from the "present time." Such a lag of information is re-

[1] Since the main focus of this paper is on audio-to-audio alignment, the alignment from audio to score or other symbolic format is not in the scope of this paper. Note that an audio-to-score alignment task is usually done through an audio-to-audio alignment task, in which the reference MIDI is synthesized into an audio signal [12].

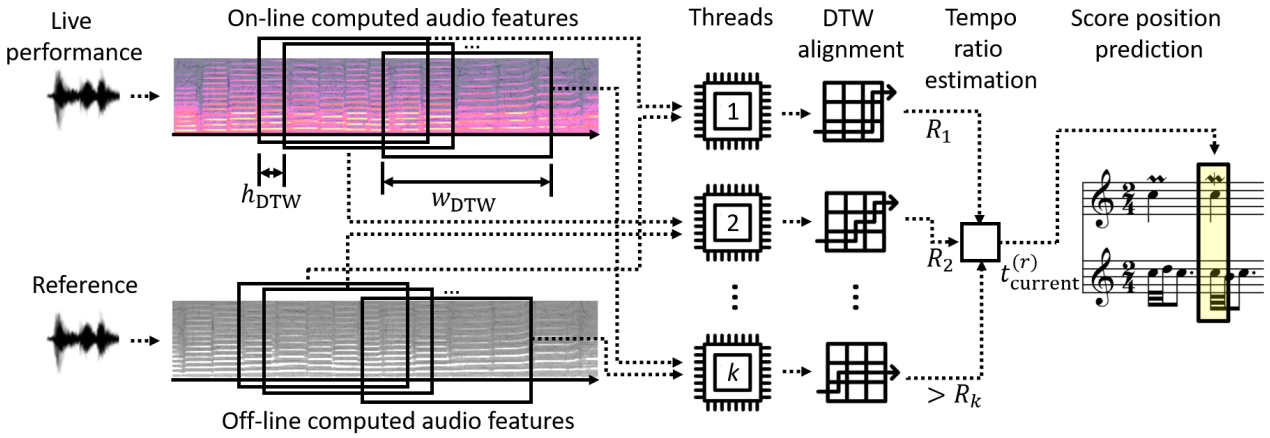

Fig. 1: Conceptual illustration of the parallel dynamic time warping (PDTW) system for on-line music alignment.

ferred to as the intrinsic latency. The longer the intrinsic latency is, the more the prediction deviates from the ground truth. The intrinsic latency is independent from the hardware speed, and the only way to reduce the intrinsic latency is to reduce the size of the window. However, a too short segment tends to give inaccurate estimation of tempo.

- Time complexity and computing speed. The time complexity of a DTW algorithm is $O(N^2)$, where N (i.e., number of points) of the input audio. The actual computational time of a segment-level DTW also varies, and tends to cause extra instability in tempo estimation. To reduce the time complexity of DTW, variants of DTW have also been proposed, such as FastDTW [19], Online DTW (ODTW) [15] and windowed time warping (WTW) [16], by adding more constraints or by omitting the backtracking process.

- Alignment accuracy. For interactive performance, the ability to make prediction precisely about future events is crucial. Since the intrinsic latency and the computational time are inevitable, calibrating the latency while keeping the predicted score position accurate relies on the prediction algorithm, such as Hidden Markov Model (HMM) [9], particle filters [10], and Monte-Carlo inference [20], have also been adopted in previous studies.

Although audio-to-audio alignment algorithms for offline cases are well developed such as Fast-DTW [19], windowed time warping (WTW) [16], HMM [9], particle filters [10], Monte-Carlo inference [20], as well as the multi-core approach [11, 21], there are still issues not fully addressed by these method in online scenarios. For example, [15, 16] can shorten the tracking response time but sacrifice the accuracy. [11] require multi-track information to launch the multi-thread architecture. [21] has data transfer overhead issue between CPU and GPU and therefore is not directly applicable for online scenarios which need fast response. [9, 10, 20] need accurate note onset information of the reference signal to utilize the score state.

The idea of utilizing an ensemble of audio to improve the performance of alignment has emerged recently. In [14], different versions of performance acting as the reference signal can enhance alignment accuracy without detailed annotation to every performance. In [11], parallel computing is used to align individual instrument part of polyphonic signals. Being inspired by these two works, we propose the PDTW algorithm as a new solution to address the issues mentioned above in interactive applications.[2]

3. MULTI-CORE ALIGNMENT SYSTEM

3.1. Parallel Dynamic Time Warping (PDTW)

Fig. 1 illustrates the working flow of the proposed PDTW system. First, the *reference* audio signal is synthesized from a MIDI sequence which represents the ground truth. This reference audio is employed to align with the audio signal of the live *performance*. To facilitate the discussion, we denote the time index of the reference, namely the *reference time*, as $\mathcal{T}^{(r)} := [t_1^{(r)}, t_2^{(r)}, \cdots, t_i^{(r)}, \cdots, t_I^{(r)}]$, $i \in [1, I]$, where $t_i^{(r)}$ is the i-th index of the reference time. Similarly, for the live performance, the *performance time* is denoted as $\mathcal{T}^{(p)} := [t_1^{(p)}, t_2^{(p)}, \cdots, t_j^{(p)}, \cdots, t_J^{(p)}]$, $j \in [1, J]$, where $t_j^{(p)}$ is at the j-th frame of the performance. To further clarify the timing relationship between live performance audio and reference audio, we also show it in Fig. 2.

In the preprocessing stage, we convert both the reference and the performance audio signals into spectral features. The

[2]Hands on demo application program is available and can be found at https://github.com/Sma1033/Realtime-audio2audio-alignment

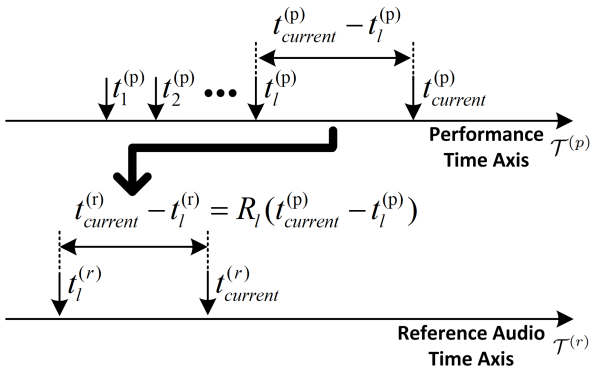

Fig. 2: The process of predicting current reference audio position using extrapolation.

features are then divided into multiple overlapped *segments* with length w_{dtw}, and the step size between every two consecutive segments is h_{dtw}. Next, a new subroutine is created to align the newest performance segment with the newest reference one using the DTW algorithm. The average slope of the resulting DTW alignment path is then the *tempo ratio* R, representing the ratio of the two instantaneous tempi of the reference and of the performance. Suppose the DTW alignment path is denoted by $f : \mathcal{T}^{(p)} \rightarrow \mathcal{T}^{(r)}$, meaning that $t_i^{(r)} = f(t_j^{(p)})$ for some i and j. The tempo ratio of the lth segment R_l is then computed by averaging the local derivatives of the DTW alignment path:

$$R_l := \frac{1}{w_{\text{dtw}} - d} \sum_{m=d+1}^{w_{\text{dtw}}} \frac{f(t_m^{(p)}) - f(t_{m-d}^{(p)})}{t_m^{(p)} - t_{m-d}^{(p)}} . \qquad (1)$$

In this paper we set $d = 10$ time indexes, as our pilot study shows that this setting yields a stable estimation of tempo ratio. Finally, for every k most recently estimated values of tempo ratio, the corresponding reference audio position $\tilde{t}_{\text{current}}^{(r)}$ is computed simply by linear extrapolation:

$$\tilde{t}_{\text{current}}^{(r)} = \frac{1}{k} \sum_{l=1}^{k} \left(t_l^{(r)} + R_l (t_{\text{current}}^{(p)} - t_l^{(p)}) \right) , \qquad (2)$$

where $t_{\text{current}}^{(p)}$ is the current performance time (i.e., the time that the live performance goes) and $\tilde{t}_{\text{current}}^{(r)}$ is the *estimated* reference time that corresponds to $t_{\text{current}}^{(p)}$. In other words, with 1) the reference audio time of the last time step given by the previous DTW and 2) the tempo ratio of the current performance time given by the current DTW, we can estimate the corresponding reference time for each single time instant. We create k threads to process the highly overlapped segment with such subroutines in parallel. By averaging the most recent estimations obtained from the k threads, $\tilde{t}_{\text{current}}^{(r)}$ is estimated in a stable manner.

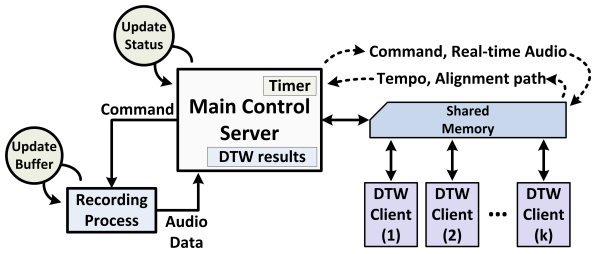

Fig. 3: System architecture of PDTW.

3.2. System Architecture

Fig.3 shows the proposed multi-thread PDTW system architecture. The system-level information and the decision-making process (i.e., Eq. (2)) are handled by the *main control server*. When the system is turned on, k clients (i.e., threads) are initialized.[3] The incoming audio data is saved in a buffer, and is periodically updated every 50 ms by the audio interface (i.e., the sound card or any hardware that receives the audio in the environment). The main control server periodically checks the status of each DTW client every h_{dtw} second. Then the server randomly assigns the latest received data from the audio interface to one of the idle clients for DTW computing. When there is a DTW alignment task completed, the server updates the tempo estimation accordingly through Eq. (1).

4. EXPERIMENTS AND EVALUATION

4.1. Dataset

In order to evaluate the performance of the proposed algorithm, we collect 10 music clips from the resource including the Bach10 dataset [22], the MAPS dataset [23], the Hainsworth dataset [24] and two classical pieces of Mozart and Schubert recorded by our collaborators (two professional musicians). The lengths of the audio files range from 1'31" to 5'09" and are sampled at 22,050 Hz. To study the performance of the system under the circumstances of highly-varying tempo values, we keep only the first 15 seconds of every music piece at the original playback speed, and then gradually change the tempo over time in an arbitrary manner, up to a minimum tempo ratio at 0.5 and a maximum one at 1.8. This results in an average value of 7.48% playback speed change in one second for stretched audio. Time-stretched audio files are generated using the TSM toolbox [25] since it allows customized path stretching.

[3]The number of clients needs not to be the same as the number of segments. However, to simplify the discussion, we assume they to be the same in this paper and denote this number as k.

4.2. Experimental settings

We test the proposed algorithm on a consumer PC with Intel i7 7700K CPU and 32GB Memory. The CPU frequency is set at 4.5 GHz, with Hyper-Threading activated. The peak double precision floating point performance of this machine is about 144 GFlops (36 GFlops / core). For audio interface, We use Yamaha MG-10XU for receiving the live audio. The input buffer is set to 0.05 sec. For feature representation, we adopt the HCQT feature proposed in [26], where we concatenate the channel-wise HCQT in different bands, with frequency from 64 Hz to 1.3 kHz, into a feature vector.

Similar to most of the audio-to-audio alignment systems [27], we evaluate the performance in frame-level. First, we introduce the concept of *tolerance window* δ. Assume there are M frames in the dataset. An estimated value $\tilde{t}^{(r)}$ of frame r is said to be a true positive if $|\tilde{t}^{(r)} - t^{(r)}| < \delta$. The accuracy (ACC) is defined as the ratio between the number of true positive frames to all the M frames:

$$\text{ACC}(\delta) = \frac{\#.\text{ true positive frames}_{\delta}}{M} \tag{3}$$

The mean error (ME) is the average deviation of detection from the ground truth:

$$\text{ME} = \frac{1}{M}\sum_{m=1}^{M}\left|\tilde{t}_m^{(r)} - t_m^{(r)}\right| \tag{4}$$

Then, we employ a procedure similar to [27] to measure system agility. We define *latency* as the average time interval between tempo update and receiving live performance signal. Finally, to evaluate computational resource usage, we record *CPU usage* through HWiNFO64[4].

4.3. Evaluation of system performance

First, we investigate the performance of the PDTW and the conventional single-thread DTW. Fig. 4a compares the resulting ACCs using different number of threads $k = 1$ (single-thread), 2, 4, and 12. All experiments are with $w_{\text{dtw}} = 6$ secs and $h_{\text{dtw}} = 0.15$ sec. Clearly, PDTW using multi-thread processing yields much better ACC than using single-thread processing. For example, when $\delta = 250$ ms, all the three resulting ACCs with $k > 1$ are by 10% higher than the one with $k = 1$. The ACC also increases consistently for $\delta > 500$ ms, meaning that multi-threading greatly helps for those segments with large alignment errors.

Fig. 4b shows the ACCs with varying DTW alignment window w_{dtw}, and with h_{dtw} fixed to 0.15 second. For more baselines of comparison, we also list the results two degenerate cases, one is simply the off-line DTW which represents the upper limit of ACC, and the other, which represents the lower limit of ACC, is using fixed tempo found at the beginning of the music piece to alignment the whole piece. The errors in the off-line DTW is mainly caused by the non-ideal properties of the time-stretching operation such as distortion of timber. Therefore, such results are reasonable and suitable for further comparison. Other results indicate that the ACC values of PDTW heavily depend on w_{dtw}. This is because short DTW alignment result can provide better tempo estimation in the local region, and such estimation can be better stabilized through multiple DTW estimations, especially when tempo change occurs.

Fig. 4c shows the ACCs with different DTW hop sizes h_{dtw}. It can be seen that the ACC is in general higher for smaller h_{dtw}. The only exception is the case of h_{dtw} = 100 ms, where its ACC is lower then the case of h_{dtw} = 500 ms for $\delta >$ 1000 ms. The reason is that when h_{dtw} is low, the system tends to localize the information within relatively stable tempo up to a very accurate position. However, when there is a high tempo variation, the system falls short of catching long-term information, and is likely to 'get lost' in this case, thereby increases the alignment error.

Finally, Fig. 5 shows the latency, ME and CPU usage using $w_{\text{dtw}} = 6$ seconds for different h_{dtw}. For a relative large h_{dtw}, ME decreases along with latency when we decrease h_{dtw}. However, if h_{dtw} is reduced to less than 0.50 second, ME increases greatly, because the system tends to overlook long-term information and therefore unfeasible for the audio with high tempo variation. Such as phenomenon can also be seen in Fig. 4c. Since what we consider is not merely the performance in the local region but in long-term one, this issue requires extra effort to be addressed in future work.

4.4. Application in live performance

The proposed PDTW alignment algorithm has been applied in an enriched music performance. [5, 6]. The event was held in the National Concert Hall of Taiwan, a concert hall with more than 2,000 seats in a 90m*60m area. Fig. 6 shows the scene viewed from the control panel room in the concert hall (left), and an illustration of piano-roll rolling through the screen along with the performance (right). In the concert, we connected a PDTW system to a music visualization interface having visual designs of musical concepts, including piano-rolls, instrument activation, and the tonnetz representation [28]. The music visualization interface retrieves the latest reference time $\tilde{t}^{(r)}_{\text{current}}$ outputted by the PDTW system every 50 ms, and illustrates the musical concepts corresponding to that time instance accordingly. Two music pieces were performed with the visualization: Mozart's *Divertimento*, K. 136, and Schubert's *Arpeggione Sonata*, D. 821. The former piece has a steady tempo, while the latter one contains huge variations of speed. Our proposed PDTW system was capable of ensuring accurate estimation of the instantaneous tempo as well as the alignment result for both music piece without prior knowledge about the acoustic response of the concert hall and the preferred interpretation of the performer.

[4]system monitoring software HWiNFO64: https://www.hwinfo.com/

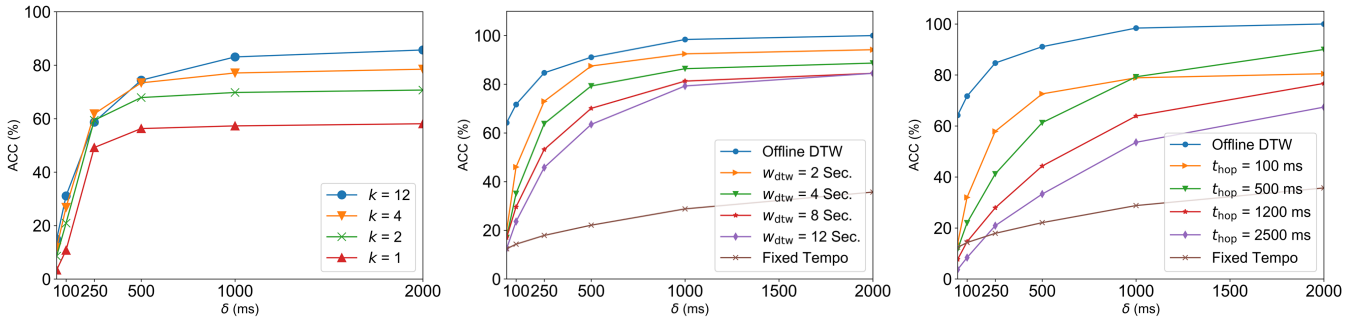

Fig. 4: (a) Comparison of single-thread DTW and PDTW. (b) Comparison of ACC with different DTW window size (w_{dtw}) and tolerance window (δ) in PDTW. (c) Comparison of ACC with different DTW hope size (h_{dtw}) and tolerance window (δ) in PDTW.

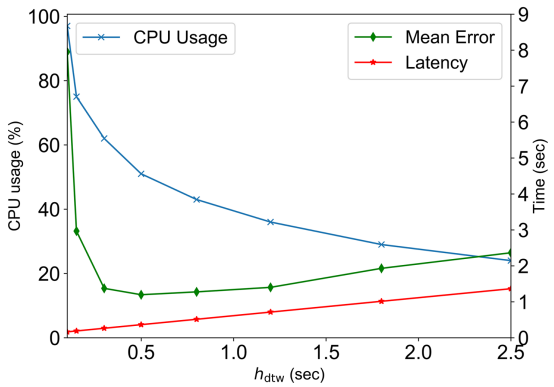

Fig. 5: CPU usage, mean error and latency with different h_{dtw}.

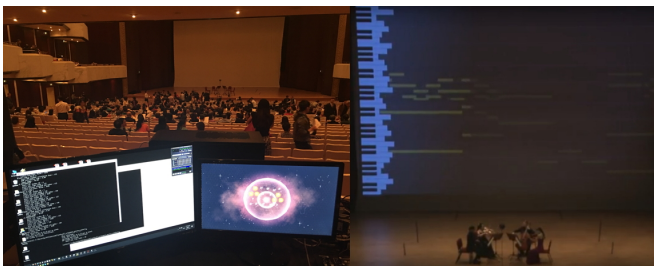

Fig. 6: Pictures taken in an enriched musical concert using PDTW algorithm for online alignment. The played notes, instrument activation and tonnetz diagram are projected on the screen behind the stage.

5. DISCUSSION AND CONCLUSION

We have presented the feasibility of the proposed PDTW algorithm for online audio-to-audio music alignment. Together with visualization, the proposed PDTW alignment algorithm has been successfully demonstrated in an enriched music performance. Besides, through a systematic evaluation, we suggest that, for a standard PC, say, one with a 4-core CPU, setting h_{dtw} = 0.5 sec, w_{dtw} = 5 secs, and 4 DTW clients would perform in a stable manner. If there are more than 8 cores available on the platform, the h_{dtw} value can be further reduced to enhance alignment accuracy. Besides its better performance in terms of accuracy and latency, another reason to advocate more research efforts on multi-thread processing is that it is inherently a more suitable architecture for the development of interactive system. Assigning subtasks simultaneously makes it much easier than assigning them sequentially to design the pipeline. With such design consideration, it's plausible to run an interactive system with more information processed simultaneously and thus can create more possibilities on the interaction between machines and musicians.

6. ACKNOWLEDGEMENT

This research was supported by the Data Science Seed Research Project of Academia Sinica. The authors thank Sheau-Jwu Wu for the visual design and art administration, the Pacing Art Culture Education Foundation for supporting the concert, and the National Concert Hall for providing the venue.

7. REFERENCES

[1] T.-C. Yeh, H.-H. Li, and J.-S. R. Jang, "Automatic hit time generation for music rhythm games," in *Proc. ISMIR*, 2014.

[2] J. Jaime, I. Barbancho, C. Urdiales, L. J. Tardón, and A. M. Barbancho, "A new multiformat rhythm game for music tutoring," *Multimedia Tools and Applications*, vol. 75, no. 8, pp. 4349–4362, 2016.

[3] J. M. Sánchez-Jara, "Digital schola: music readers as learning/teaching tools," in *Proc. 2nd Int. Conf. Technological Ecosystems for Enhancing Multiculturality*, 2014, pp. 547–553.

[4] R. B. Dannenberg and C. Raphael, "Music score alignment and computer accompaniment," *Communications of the ACM*, vol. 49, no. 8, pp. 38–43, 2006.

[5] E. Maestre, P. Papiotis, M. Marchini, Q. Llimona, O. Mayor, A. Pérez, and M. M. Wanderley, "Enriched multimodal representations of music performances: Online access and visualization," *IEEE MultiMedia*, vol. 24, no. 1, pp. 24–34, 2017.

[6] C. C. S. Liem, E. Gómez, and M. Schedl, "PHENICX: Innovating the classical music experience," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2015)*, Torino, Italy, June–July 2015.

[7] L. Turchet, C. Fischione, and M. Barthet, "Towards the internet of musical things," in *Proc. Sound and Music Computing (SMC)*, 2017.

[8] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, "Evaluation of real-time audio-to-score alignment," in *Proc. ISMIR*, 2007.

[9] A. Cont, "Realtime audio to score alignment for polyphonic music instruments, using sparse non-negative constraints and hierarchical HMMs," in *Proc. ICASSP*, 2006, vol. 5.

[10] T. Otsuka, K. Nakadai, T. Takahashi, T. Ogata, and H. G. Okuno, "Real-time audio-to-score alignment using Particle Filter for coplayer music robots," *EURASIP Journal on Advances in Signal Processing*, p. 2, 2011.

[11] P. Alonso, R. Cortina, F. J. Rodríguez-Serrano, P. Vera-Candeas, M. Alonso-González, and J. Ranilla, "Parallel online time warping for real-time audio-to-score alignment in multi-core systems," *The Journal of Supercomputing*, vol. 73, no. 1, pp. 126–138, 2017.

[12] F. J. Rodriguez-Serrano, J. J. Carabias-Orti, P. Vera-Candeas, and D. Martinez-Munoz, "Tempo driven audio-to-score alignment using spectral decomposition and online Dynamic Time Warping," *ACM Trans. on Intelligent Systems and Technology*, vol. 8, no. 2, pp. 22, 2016.

[13] A. Arzt and G. Widmer, "Simple tempo models for real-time music tracking," in *Proc. Sound and Music Computing (SMC)*, 2010.

[14] A. Arzt and G. Widmer, "Real-time music tracking using multiple performances as a reference.," in *Proc. ISMIR*, 2015, pp. 357–363.

[15] S. Dixon, "Live tracking of musical performances using On-line Time Warping," in *Proc. the Int. Conf. DAFx*, 2005, pp. 92–97.

[16] R. Macrae and S. Dixon, "Accurate real-time Windowed Time Warping.," in *Proc. ISMIR*, 2010, pp. 423–428.

[17] B. Lehner, G. Widmer, and S. Bock, "A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks.," in *Proc. EUSIPCO*, 2015, pp. 21–25.

[18] L. Su and H.-t. Wu, "Minimum-latency time-frequency analysis using asymmetric window functions," *arXiv preprint arXiv:1606.09047*, 2016.

[19] S. Salvador and P. Chan, "FastDTW: Toward accurate Dynamic Time Warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[20] N. Montecchio and A. Cont, "A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Monte Carlo inference techniques," in *Proc. ICASSP*, 2011, pp. 193–196.

[21] L. Xiao, Y. Zheng, W. Tang, G. Yao, and L. Ruan, "Parallelizing Dynamic Time Warping algorithm using prefix computations on GPU," in *IEEE 10th International Conference on High Performance Computing and Communications*, 2013, pp. 294–299.

[22] 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, 2010.

[23] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Trans. Audio, Speech, Lang. Proc.*, vol. 18, no. 6, pp. 1643–1654, 2010.

[24] G. Tzanetakis and G. Percival, "An effective, simple tempo estimation method based on self-similarity and regularity," *Proc. ICASSP*, pp. 241–245, 2013.

[25] J. Driedger and M. Müller, "TSM Toolbox: MATLAB implementations of time-scale modification algorithms," in *Proc. Int. Conf. DAFx*, Erlangen, Germany, 2014, pp. 249–256.

[26] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 estimation in polyphonic music," in *Proc. ISMIR*, 2017.

[27] A. Cont, "A coupled duration-focused architecture for real-time music-to-score alignment," *IEEE Tran. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 974–987, 2010.

[28] L. Bigo, D. Ghisi, A. Spicher, and M. Andreatta, "Representation of musical structures and processes in simplicial chord spaces," *Computer Music Journal*, vol. 39, no. 3, pp. 9–24, 2015.