Robust and Efficient Joint Alignment of Multiple Musical Performances

Siying Wang, *Student Member, IEEE,* Sebastian Ewert, *Member, IEEE,* and Simon Dixon

***Abstract*—The goal of music alignment is to map each temporal position in one version of a piece of music to the corresponding positions in other versions of the same piece. Despite considerable improvements in recent years, state-of-the-art methods still often fail to identify a correct alignment if versions differ substantially with respect to acoustic conditions or musical interpretation. To increase the robustness for these cases, we exploit in this work the availability of multiple versions of the piece to be aligned. By processing these jointly, we can supply the alignment process with additional examples of how a section might be interpreted or which acoustic conditions may arise. This way, we can use alignment information between two versions transitively to stabilize the alignment with a third version. Extending our previous work [1], we present two such joint alignment methods, progressive alignment (PA) and probabilistic profile (PP), and discuss their fundamental differences and similarities on an algorithmic level. Our systematic experiments using 376 recordings of 9 pieces demonstrate that both methods can indeed improve the alignment accuracy and robustness over comparable pairwise methods. Further, we provide an in-depth analysis of the behaviour of both joint alignment methods, studying the influence of parameters such as the number of performances available, comparing their computational costs, and investigating further strategies to increase both their computational efficiency and alignment accuracy.**

***Index Terms*—Music synchronization, multiple sequence alignment, performance analysis, robust music alignment.**

I. Introduction

DURING the last decades alignment methods in various forms have been of central importance for the analysis, modelling and processing of digital music recordings. By establishing links between similar sections across different recordings or representations of music, alignment techniques enable a multitude of applications, including automatic score following and page turning [2], [3], facilitated navigation in large collections [4], the identification of cover songs [5], query-by-example retrieval [6] and the integration of prior knowledge in audio source separation [7].

In this paper, we focus in particular on the task of *music alignment*, or synchronisation, which is given a position in one version of a piece of music, to locate the corresponding position in another version. In this context, various alignment methods have been proposed, including Dynamic Time Warping (DTW) [8], Hidden Markov and Semi-Markov Models (HMM) [9], Conditional Random Fields (CRF) [10], general graphical models [11], and Particle Filter / Monte-Carlo Sampling (MCS) based methods [3], [12]. As shown in previous studies, current methods based on such synchronization strategies yield alignments of high accuracy in many cases [10], [13], [14]. However, a musician can interpret a piece in diverse ways, which can lead to significant local differences between versions in terms of articulation and note lengths, ornamental notes (grace notes, trills), or the relative loudness of notes (balance). Additionally, substantial differences in the acoustic environment, instrumentation and recording conditions can reduce the alignment accuracy of state-of-the-art methods drastically.

To improve the alignment accuracy for such difficult cases, a recently presented concept exploits the fact that in many cases not only two but multiple versions of a piece are available [1]. This is the case, for example, in comparative performance analysis [15]–[17] and expressivity studies [18], in music production where corresponding audio takes need to be aligned [19], when coordinating user-generated videos of a concert [20] or generating ground-truth for large scale distance learning [21]. If multiple versions are indeed available, the idea is to align them in a joint way, which facilitates the synchronization process as every additional version presents another example of how a musician can realize a section of a piece or which acoustic conditions might prevail in a recording. Fig. 1 shows a real-world example of a pair-wise method failing to compute a correct alignment between two recordings of Chopin's Op. 24 No. 2. Chroma features (see [22], [23] for a detailed description) for both recordings shown in Fig. 1b and c reveal acoustical (more noise in the C♯ and D chroma bands in Fig. 1c) and musical differences (more pronounced staccato on the E and G notes in Fig. 1b). Since the piece shows a repetitive pattern on the chroma level, such differences cause a pairwise method [13] to compute an incorrect alignment between the two versions. The results are shown in Fig. 1d as a gray alignment path, which encodes corresponding positions between the two recordings as computed by the method – note how the path deviates from the correct positions between 57.5–61.5 seconds (in the timeline of Luisada's version). However, as we will see, including several recordings in a joint alignment process can lead to a considerable increase in overall robustness and alignment accuracy. For now, we only indicate this improvement showing two additional paths (dashed and dotted) in Fig. 1d that were computed using our joint alignment methods.

In general, a joint synchronization of music recordings can be considered as an instance of the *multiple sequence alignment problem*, a task well-studied in bio-informatics [24], [25]. In this context, the approach presented in [1] belongs to the class of *progressive alignment (PA) methods*. A

The authors are with the Centre for Digital Music, School of Electrical Engineering and Computer Science, Queen Mary University of London, London, U.K. (e-mail: {siying.wang,s.ewert,s.e.dixon}@qmul.ac.uk.)

Manuscript received Dec 30, 2015; revised July 17, 2016.

second class used in bioinformatics is commonly referred to as *probabilistic profile (PP) methods*. As discussed in more detail below, methods from both classes typically share specific algorithmic roots, with some conceptual differences in the internal representation of the sequences to be aligned. In particular, PA methods typically employ a more greedy, quickly converging approach compared to PP methods, which often leads to a considerably higher computational efficiency for the former. However, as reported in [26], PP methods were found to yield a higher alignment accuracy in some bioinformatics tasks. Music recordings, however, have properties quite different from protein sequences. First, we do not consider structural differences between performances (e.g. a section being left out in one version), which is in stark contrast to biological sequences where such fundamental differences are common. Second, music recordings change more slowly over time leading to a high temporal correlation between neighboring sequence elements, which again is quite different from protein sequences. Due to these differences, it is interesting to test whether a higher alignment accuracy can also be achieved in a musical context using PP methods and how the two methods differ in behavior in such a scenario.

As a first main contribution of this paper, we compare the method presented in our previous work [1] with a PP method we develop for application to music. We conduct systematic experiments to identify their conceptual advantages and disadvantages in different scenarios. To increase the comparability between the two approaches, we employ the same feature types and configurations in both cases. As a second main contribution, we provide additional insights into the behaviour of each joint alignment method by illustrating the influence of individual parameters with associated detailed experiments, and describe extensions to accelerate the PP method and improve the alignment accuracy of the PA method.

The paper is organized as follows. We discuss related work in Section II. Technical details of the two proposed methods are described in Section III, followed by a systematic comparison in Section IV. In Section V, we report on the results of our in-depth investigations of the two methods. Conclusions and discussions of future work are given in Section VI.

II. Related Work

Music synchronization has been an active research topic for several decades. Early approaches [28], [29] are based on string matching algorithms, that were used to align a symbolic music representation, e.g. MIDI, with a given score. Since the 1990s, the increase in computing power enabled the processing of audio signals, and efforts have shifted towards robust feature representations and suitable alignment methods for aligning audio recordings. For the feature representation, a major aim is to find an optimal, application-specific trade-off between the level of detail preserved in a feature and its robustness against noise and other musically irrelevant signal properties. In this context, low-level spectral representations have been used [30]–[32] as well as musically meaningful representations, especially pitch and chroma features [27], [33], [34]. More recently, it was found that accompanying

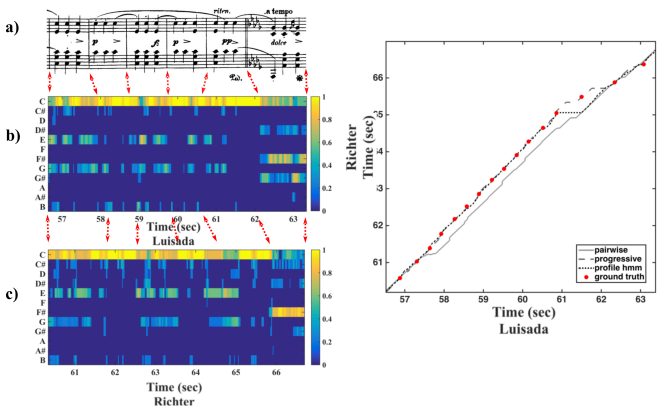

Fig. 1. Alignment of two interpretations of Chopin Op. 24 No. 2, measures 52-57: **(a)** Score for the six measures. **(b)/(c)** CENS features (a variant of chroma features proposed in [27]) for an interpretation by Luisada and Richter, respectively. **(d)** Alignment results for the pairwise (blue), proposed progressive alignment (black) and profile HMM (green); ground truth are given as corresponding beat positions from the two versions (red).

such representations with features indicating onset positions can be used to improve the alignment accuracy [10], [13]. Other more recent developments are adaptive or learnt feature representations [21], [34]–[36].

Once a common feature representation is chosen, the feature sequences are aligned using a suitable method. While there are many possible ways to classify methods, the most distinctive differences can be found between offline and online/realtime methods. In particular, almost all offline methods are based on dynamic programming (DP). For example, techniques based on Dynamic Time Warping (DTW) aim at finding an alignment minimizing the dissimilarity between features assigned to each other – effectively a constrained optimization problem that can be solved using DP [37], [38]. Early examples of such methods exist which use low-level [30] and mid-level [33] features.

Interpreting the synchronization problem as a latent state estimation problem leads to Hidden Markov Models (HMMs) in their various forms [9], [32], [34], [39], [40]. Such models have been particularly popular in score-to-audio alignment tasks, as here each state intuitively corresponds to a note or a constellation of concurrent notes as specified by the score, while other assumptions about the music can be captured in higher-level HMM structures. For examples, high-level states might encode the current tempo [11], or each note-state can be subdivided into attack-decay-release sub-states (or similar temporal evolutions). Such high level structures lead to hierachical HMMs and semi-Markov graphical models, or generalizations thereof such as Dynamic Bayesian Networks (DBNs) [32], [41]. It should be noted that many of these more advanced models can still be represented as a standard HMM. Recently, conditional random fields (CRFs) have been used for music synchronization [10]. As an advantage, CRFs loosen several limitations of HMM-based methods in contrast to more general DP methods, e.g. DTW. In particular, their use of so-called feature-functions generalizes the notion of observation probability and thus enables measuring distances between features in a more general way than HMMs allow.

Several strategies have been proposed to lower the computational costs for techniques based on dynamic programming. In [42], [43], first a rough alignment is computed on a low temporal resolution, which is then used to constrain the alignment process at higher resolutions – effectively a path pruning technique suitable for offline methods. Many other techniques not only accelerate but enable a method to align sequences online or in real-time. For example, the method presented in [14] employs a greedy, locally optimal forward path estimation algorithm to constrain the alignment path, while [44] employs a windowed variant of DTW integrating ideas of the A^* algorithm [45] to dynamic programming. Such methods are combined in [2] with an indexing system to efficiently establish a first, approximate score-to-audio positioning, which is able to handle performances starting from arbitrary positions in a piece. Conceptually quite different from the above are state-space methods, where states such as position or tempo are elements of a continuous space. Transitions between states are modelled using transition functions that, applied to the current state, yield the next one [12]. Depending on specific properties of the sources of noise in the model, one typically uses parameter estimation methods based on the Kalman filter, particle filter or more general Monte-Carlo sampling methods.

Aligning multiple performances of a piece of music is a relatively novel concept in music processing. A generative note duration model is proposed in [46] by coupling the tempo curve from different audio performances. The method in [47] uses multiple performances to improve the accuracy in an on-line score-following application by computing several pairwise alignments in parallel. Further, in [48] the authors align multiple symbolic sequences for harmonic and motivic analysis.

In some sense, our idea of using multiple versions to improve the performance of music alignment is similar to the co-segmentation problem in computer vision, where the segmentation accuracy can often be improved by supplying the algorithm with additional images that share certain foreground characteristics with a given image and segmenting them jointly [49]. More directly relevant to our work, however, are multiple sequence alignment methods, which have been of central importance to many developments in bioinformatics. For example, the MAFFT [50] and CLUSTAL [51] families of algorithms have been in development for almost three decades. While a lot of the functionality in such packages is highly specific to the alignment of protein sequences, we can extract some central ideas and adapt them suitably, taking music specific properties into account. In the next section we describe two such adapted methods: progressive and profile-based alignment and discuss some conceptual differences between them.

III. Methods for Joint Music Alignment

In theory, it is straightforward to extend many dynamic programming techniques to multiple dimensions so that several sequences can be aligned jointly. For example, this has been demonstrated in the context of gesture recognition [52] and multi-modal speech recognition [53]. However, assuming that each sequence to be aligned is roughly of length N, the time and memory requirement to align K versions is $O(N^K)$, which limits K to very small values in practice. Path pruning techniques can be used to mitigate such problems for small values of K [54], but in general for large K, it can be very difficult to lower the computational costs enough to become practically feasible and find accurate alignments at the same time.

A different strategy is to successively build up a data structure representing an average sequence or *central consensus* against which all given sequences can be aligned. By constructing this consensus form, we can incorporate information from every single recording such that the overall alignment becomes easier (as the influence of outlier information can be reduced) and therefore becomes more accurate and robust. In the following, we present two conceptually different methods for computing such a central consensus in a music synchronization scenario. The differences include how the central consensus is represented (keeping all information in contrast to averaging some) and how it is built up (early versus late updates). Both of these affect the resulting alignment accuracy and computational performance, as we discuss in more detail below.

A. Progressive Alignment

Our first method can be regarded as a member of the family of progressive alignment algorithms in the context of bioinformatics [50]. As not all of its steps are directly interpretable from a probabilistic point of view, we present the method as a general dynamic programming approach. The idea is, instead of simultaneously aligning all feature sequences, to successively add the sequences to a data structure referred to as the *template*. The template comprises a set of feature sequences that are aligned to each other, stretched in length to have the same size – as we will see, this enables efficient access to aligned sequence elements. After computing an alignment between a new sequence and the template at each step, the sequence is added using the alignment information to stretch both the template and the sequence to have the same size. This is repeated until all sequences are contained in the template, which allows for efficiently deriving pairwise alignments between any two sequences.

To describe the alignment procedure in more detail, we assume that we have K different versions of a piece and that their feature sequences are denoted by $X^k = (x_1^k, \ldots, x_{N_k}^k)$ with $k \in [1:K]$ and $x_n^k \in \mathcal{F}$, where $\mathcal{F}$ denotes a suitable feature space. Further, we refer to our template data structure as Z, which we initialize to X^1. As part of the alignment process, we align the remaining feature sequences $X^2, \ldots, X^K$ successively to Z, updating Z after each step. To this end, let X^k denote the sequence to be aligned and $Z = (z_1, \ldots, z_M)$ the current template of length M. Each $z_m \in (\mathcal{F} \cup \{G\})^{k-1}$ contains $k-1$ feature vectors or *gap symbols* G, and we denote the individual components by $z_m^1, \ldots, z_m^{k-1} \in \mathcal{F} \cup \{G\}$. The idea behind the gap symbol will be discussed below. Further, to simplify the notation later, we denote the sequence $(z_1^r, \ldots, z_M^r)$ by Z^r for $r \in 1, \ldots, k-1$.

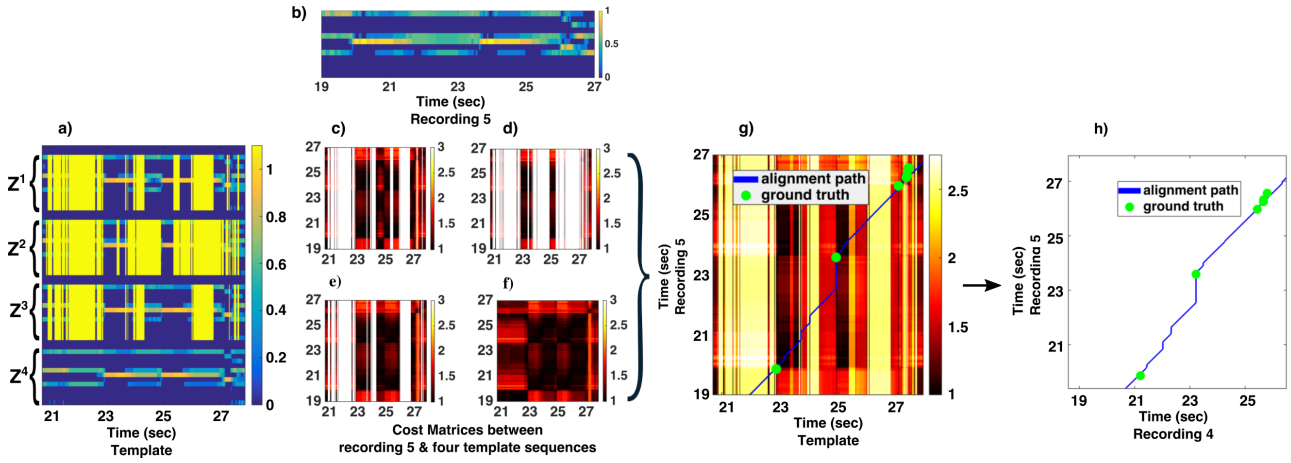

Fig. 2. Aligning the fifth recording with the template of the first four recordings: we compare the feature sequence of the fifth recording (b) with each feature sequence in the template (a: yellow blocks indicate the gaps) to obtain four cost matrices (c,d,e,f); we combine these cost matrices together into a single cost matrix to compute the alignment (g: the ground truth onset position for the alignment between the fourth and the fifth recording is stretched according to the gap inserted version of the fourth feature sequence, in order to be fitted onto the cost matrix between the template and the fifth recording). The resulting alignment path between the fourth and fifth recordings is shown in (h).

An alignment between Z and X^k is defined as a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (m_\ell, n_\ell) \in [1:M] \times [1:N_k]$ for $\ell \in [1:L]$ satisfying $1 = m_1 \leq m_2 \leq \ldots \leq m_L = M$ and $1 = n_1 \leq n_2 \leq \ldots \leq n_L = N_k$ (boundary and monotonicity conditions), as well as $p_{\ell+1} - p_\ell \in \{(1,1),(1,0),(0,1)\}$ (step size condition). To compute an alignment p between Z and X^k, we first define a dissimilarity measure for individual elements from Z and X^k. More precisely, we compute a cost matrix $C^r \in \mathbb{R}^{M \times N_k}$ comparing each pair of elements in Z^r and X^k, by:

$$C^r(m,n) = \begin{cases} c(z^r_m, x^k_n), & z^r_m \neq G, \\ \mathcal{C}_G, & z^r_m = G, \end{cases}$$

where $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ is a suitable dissimilarity measure between feature vectors and $\mathcal{C}_G > 0$ is a constant referred to as the *gap penalty*. By combining these individual cost matrices C^r to a *merged cost matrix* $C \in \mathbb{R}^{M \times N_k}$, we obtain a dissimilarity measure between every sequence element in Z and X^k. A simple yet effective combination is averaging:

$$C(m,n) = \frac{1}{k-1} \sum_{r=1}^{k-1} C^r(m,n).$$

This process is illustrated with an example of aligning five synthetic recordings in Fig. 2, where a template (Fig. 2a) containing four sequences of chroma-based vectors (with yellow entries indicating gap symbols) is aligned to a fifth sequence (Fig. 2b). The resulting four cost matrices C^1 to C^4 are shown in Fig. 2c-f, using $\mathcal{C}_G = 3$ and a cosine distance $c(z,x) = 2 - \frac{\langle z,x \rangle}{\|z\|\|x\|}$. The resulting merged cost matrix C is shown in Fig. 2g.

Note that we also tried other combination strategies, including weighting schemes, taking the minimum over the individual cost matrices or more general order statistics including the median and other percentiles. We also tested using logistic regression to learn a dissimilarity measure based on the individual cost matrices to optimize for overall alignment accuracy. However, using the same experimental setup as described in Section IV-A, replacing only the combination strategies, none of these strategies yielded consistently better results than the averaging described above.

Once a merged cost matrix is computed, we can apply dynamic programming similar to DTW or Viterbi decoding to derive an alignment between Z and X^k. An alignment p having minimal total cost among all possible alignments is called an *optimal alignment*. To determine such an optimal alignment, we recursively compute an *accumulated cost matrix* D of size $(M \times N_k)$, where the matrix entry $D(m,n)$ contains the total cost of an optimal alignment between $(z_1, \ldots, z_m)$ and $(x_1, \ldots, x_n)$:

$$D(m,n) := \min \begin{cases} D(m-1, n-1) + w_1 C(m,n), \\ D(m-1, n) + w_2 C(m,n), \\ D(m, n-1) + w_3 C(m,n), \end{cases}$$

for $m, n > 1$. Furthermore, $D(m,1) := \sum_{k=1}^{m} w_2 C(k,1)$ for $m > 1$, $D(1,n) := \sum_{k=1}^{n} w_3 C(1,k)$ for $n > 1$, and $D(1,1) := C(1,1)$. The weights $(w_1, w_2, w_3) \in \mathbb{R}^3_+$ can be used to adjust the preference over the three step sizes. By tracking the choice for the minimum starting from $D(M,N)$ back to $D(1,1)$, an optimal alignment can be derived in a straightforward way [55], [56].

Once an alignment p is computed, we integrate X^k into Z. To this end, we use p to stretch Z and X^k to the same length, such that corresponding features are aligned and become part of the same element of Z. There are several ways to define this stretch. A first idea is to simply set

$$\widetilde{z}_\ell = (z^1_{m_\ell}, \ldots, z^{k-1}_{m_\ell}, x^k_{n_\ell}),$$

where $\widetilde{Z} = (\widetilde{z}_1, \ldots, \widetilde{z}_L)$ denotes the updated template and $p = \big((m_1, n_1), \ldots, (m_L, n_L)\big)$. This simple solution, however, introduces a temporal uncertainty: if the step size $(1,0)$ or $(0,1)$ is used in p, an element in Z or X^k is aligned to several elements in the other sequence, respectively. Therefore, with

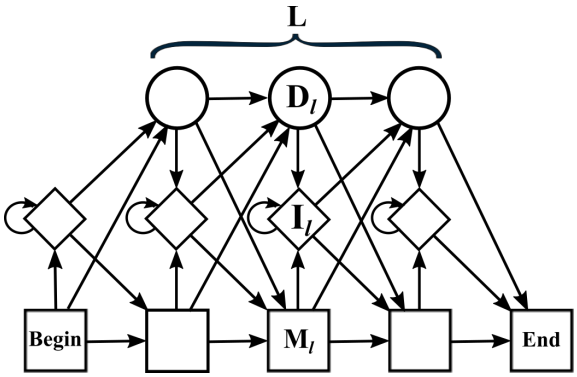

Fig. 3. Topology of a Profile HMM [24]

this simple update rule some elements of Z or X^k would occur several times in $\widetilde{Z}$ causing a temporal uncertainty, as features of the next sequence can equally well be aligned against the original or a copied feature in a template sequence.

Due to these issues, we now introduce a rule that replaces copies of elements with a gap symbol. To this end, we first define the terms

$$E_1^p(m) := \underset{\{\widetilde{n}|(m,\widetilde{n})\in p\}}{\operatorname{argmin}} C(m,\widetilde{n}),$$

$$E_2^p(n) := \underset{\{\widetilde{m}|(\widetilde{m},n)\in p\}}{\operatorname{argmin}} C(\widetilde{m},n).$$

If an element in one sequence is aligned to several in the other sequence, we can use E_1 and E_2 to find the pair of elements that has the lowest cost. With this we define our update rule as follows:

$$\widetilde{z}_\ell = \begin{cases} (z^1_{m_\ell},\ldots,z^{k-1}_{m_\ell},x^k_{n_\ell}), & \text{if } (m_\ell,n_\ell) = E_1^p(m_\ell) = E_2^p(n_\ell) \\ (z^1_{m_\ell},\ldots,z^{k-1}_{m_\ell},G), & \text{if } (m_\ell,n_\ell) \neq E_2^p(n_\ell) \\ (G,\ldots,G,x^k_{n_\ell}), & \text{if } (m_\ell,n_\ell) \neq E_1^p(m_\ell) \end{cases}$$

Intuitively, for the case that p aligns an element m of one sequence to multiple elements of the second sequence, this update rule uses C to select the best of these multiple elements to align with m, and then the remaining elements are aligned to new gap symbols. This contrasts with the simple rule where the multiple elements would be aligned to copies of m. We will investigate the importance of the gap symbol in Section V-B. Also, the order in which feature sequences are aligned is crucial to PA method, which will be discussed in Section V-C.

B. Probabilistic Profile

Another central class of multiple sequence alignment methods are probabilistic profile methods. For these methods, the central consensus data structure takes the form of a Hidden Markov Model (HMM), in a specific configuration. In the following, we describe such a *profile HMM* which we adapt for the music synchronization scenario, see also [24] for similar concepts as used in bio-informatics.

The topology of our profile HMM is illustrated in Fig. 3. Overall, the model contains three different types of states: *Match* states (M), *Insert* states (I) and *Delete* states (D). Intuitively, the series of match states will encode the core of a consensus sequence representing the commonalities among different recordings, while the insert and delete states are used to model the temporal diversity. To find meaningful parameters for the various probability distributions involved, each given sequence is interpreted as a noisy observation of the consensus sequence with insertions and deletions, and thus can be used to train the model using a Baum-Welch procedure. Interpreted in this way, it should be noted that the match states do not necessarily correspond to musically meaningful events like specific chords or note constellations as specified by a score.

To describe the model in more detail, we use the same notation as above and assume K different versions of a piece with corresponding feature sequences denoted by $X^k = (x^k_1,\ldots,x^k_{N_k})$ for $k \in [1:K]$, where each $x^k_n \in \mathcal{F}$. Further, we assume a general familiarity with HMMs and refer for details to [56]. To define the structure of the profile HMM, we first choose the length L of model: The number of M and D states is L, respectively, while there are $L+1$ I-states, compare Fig. 3. While L could simply be a constant (as often used in bio-informatics), we obtained the best results by adapting L to the length of the given feature sequences. More precisely, we set $L = \operatorname{median}(N_1,\ldots,N_K)$, which fixes the overall topology, compare Fig. 3. Other choices we tested include the minimum, maximum and twice the maximum instead of the median, but those led to a lower overall alignment accuracy in our experiments. From a generative point of view, we start from a non-emitting Begin state. From there we can enter the first match state and draw a feature vector according to the corresponding observation probability. Since match states do not have self-transitions, we either enter the second match state, enter the first insert state or the second delete state. Insert states have self transitions and thus can generate an arbitrary number of feature vectors according to their observation probability – useful for modelling observation sequences that locally have a lower tempo compared to the consensus sequence. Delete states are non-emitting states and, since transitions between them are allowed, can be used to skip an arbitrary number of match states – useful for modelling observation sequences with a higher local tempo. Note that by allowing direct jumps from a match state to subsequent, non-neighbouring match states, one could also model deletions in a different way. The separate delete states, however, are introduced to avoid the problem of specifying a maximal length for such jumps and deletions. The possible transitions are shown in Fig. 3.

To represent the observation probabilities of the match and insert states, we use multinomial Gaussian distributions with means μ^M_ℓ, μ^I_ℓ and covariance matrices $\sigma^M_\ell, \sigma^I_\ell$. A benefit of using a Gaussian distribution is that the parameters have a straightforward interpretation. In particular, the means are elements of the feature space $\mathcal{F}$ and thus the sequence $\mu^M_1,\ldots,\mu^M_L$ can be interpreted as encoding our consensus feature sequence, while the insert state means $\mu^I_1,\ldots,\mu^I_L$ encode typical features we additionally observe for recordings with a slow tempo. For the covariances we use diagonal matrices, as otherwise we would need to estimate a number of parameters for each state equal to the square of the dimension of the feature space [56], and we typically do not have enough feature sequences as training material to do so reliably. Also,

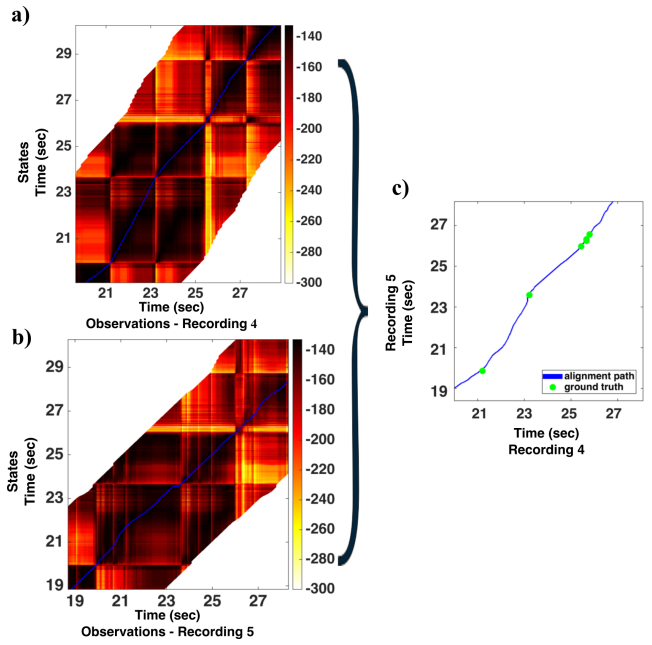

Fig. 4. Emission probability matrices for match states with alignment path (blue) between states and observations of (a) the fourth recording and (b) the fifth recording. The values on color bar are in log scale. Note that the extremely low probabilities (white area) on the top left and the bottom right corner result from the path constraint described in Section III-C; (c) The resulting alignment path between the fourth and fifth recordings.

it is reasonable to assume that the dimensions in the feature vector are roughly decorrelated in the given data, and if not this could be done as a pre-processing step. Note that instead of generating the observation model artificially from the score as the audio-score alignment task [32], we learn our consensus feature sequence from the feature sequences of all the performances. Regarding the transition probabilities, we only need to estimate a low number of parameters due to the sparsely connected structure of the profile HMM: three for each state (compare Fig. 3). Instead of fixing them to specific values, we found that estimating them from data improved the overall alignment accuracy. In particular, learnt from data, the transition probabilities encode how likely the sequences are to deviate from the consensus sequence locally, thus provide additional guidance during the alignment process.

Parameters of the model are estimated using expectation-maximization (EM), i.e. the multiple sequence variant of the Baum-Welch algorithm [56]. As in a music synchronization scenario the number of available sequences is typically several orders of magnitude smaller than in bioinformatics, the initialization strategy is crucial to avoid running into poor local maxima of the objective, i.e. the likelihood function we try to maximize using EM. We obtained the highest alignment accuracies as follows. We initialized the match and insert means using the feature vectors of a sequence having length L (as chosen above). Excluding that sequence from the training procedure enabled the model to properly account for the other sequences without overfitting the initializing sequence. Other strategies to obtain a proper initialization led to lower alignment accuracies. For example, we tried random initializations as well as resampling all sequences to the same length (corresponding to a linear stretch), followed by averaging. The covariance matrices were uniformly initialized to a fixed, relatively high value as a measure to overcome over-fitting. Additionally, to avoid the collapsing-Gaussian problem, we constrained the estimated variances to a reasonable minimum [24]. The transition probabilities were initialized uniformly, with the exception for match-match and delete-insert transitions: the former are encouraged and the latter discouraged. After training the profile HMM, we compute alignments between the model and each sequence using the Viterbi algorithm. Pairwise alignments between any two sequences can be derived using the model as a central intermediary. This last step is illustrated with an example of aligning the same recordings also used in Fig. 2. We show the observation probabilities (log-scaled) in Fig. 4a and b for two sequences against a number of match states based on the trained profile HMM. The optimal state sequence found via Viterbi decoding for each sequence is illustrated as an alignment path in white (alignments against non-match states are not directly visible in the figure and the path is interpolated accordingly). Based on the first alignment we can align a given feature vector in the first sequence to states in the profile HMM, which then can be aligned to feature vectors in the second sequence using the other alignment. The resulting pairwise alignment between the given feature sequences is illustrated in Fig. 4c. We refer for more details on the training and decoding process to [24], [56].

C. Accelerating Alignments Using Multi-Scale Dynamic Programming

To obtain alignments of high accuracy, it is necessary to use features with a high temporal resolution. The resulting increase in length of the feature sequences compared to lower resolutions, however, also leads to a considerable increase of the computational costs for the alignment methods described above. In particular, assuming that all sequences are roughly of length N, the time and memory requirements of the dynamic programming technique presented in Section III-A as well as of the Baum-Welch (in each iteration) and Viterbi algorithms used in Section III-B are quadratic in N. Therefore, for large N, the alignment problem can easily become computationally impractical or even infeasible.

To increase the computational efficiency for both joint alignment methods, we adapted the multi-scale alignment strategy proposed in the context of DTW in [42], [43]. The general idea is to recursively project a path obtained on a coarse feature resolution level to a next higher resolution and to refine the projected alignment on that level. This way, only entries (corresponding to aligned positions in progressive alignment, or feature-state combinations in the profile HMM) around the projected path in a matrix need evaluating. As shown in [42], this strategy is particularly useful for music due to the high temporal correlation between neighboring feature vectors, i.e. the temporal feature resolution can be decreased without losing the information necessary to find the correct path on the coarser level. Since progressive alignment and the Baum-Welch/Viterbi algorithms share common algorithmic roots, we

can adopt the multi-scale strategy for both methods. More precisely, similar to [13], [42], we use a total of four different feature resolutions, with the lower three ones obtained from the highest using low-pass filtering (smoothing) and down-sampling. The resulting temporal resolutions are 1 sec, 0.5 sec, 0.1 sec and 0.02 sec. After computing an alignment (or a Baum-Welch iteration) on a coarsest level, we project the path to the next finer resolution and constrain alignments to run in a neighbourhood of the projected path. The size of the neighbourhood was defined as in [13], [42]. This is illustrated in Fig. 4a and b for the profile HMM: An alignment path computed on a coarse level was projected to the final feature resolution, where it is used to constrain which entries in the observation probability matrix are computed. Entries outside the constraint region are formally given an extremely low probability (white entries). Similar constraint regions are also applied during the computation of the observation and posterior probability matrices used in the Baum-Welch algorithm. During our experiments, we observed similar speed-ups as reported in [42], i.e. the resulting methods were typically faster by a factor of 40-100 depending on the length of the recordings used – without a decrease in alignment accuracy.

IV. Comparing Pairwise, Progressive and Profile-HMM Based Alignment

The two methods described in Section III differ considerably on a formal level, with one being described as an optimization and the other as a probabilistic inference problem. On the algorithmic level, however, there are many similarities. In particular, under certain conditions, DTW is equivalent to a negative log-likelihood implementation of an HMM using multinomial Gaussian distributions for the observations, an implementation type highly advisable for HMMs due to its numerical stability [56]. This is the case if we limit ourselves to using a Euclidean distance to compare features and use additive instead of multiplicative weights for different step sizes. With these limitations, the application of a logarithm transforms the HMM-likelihood from a product of probabilities to a sum of log-probabilities, which for the case of a Gaussian takes the form of a Euclidean distance. One can show that the result is equivalent to DTW by interpreting the features of one DTW sequence as HMM states, using the features as the mean of the corresponding Gaussians and adding some non-emitting states to model certain step sizes – see [56], [57] for some discussion.

Given these algorithmic similarities between general DTW and HMM, it is interesting to note where the central conceptual differences between our two approaches are and how they could affect the alignment results. A first difference is *adaptability in size*. Our progressive method retains every feature sequence it encounters, gradually adapting the size of the template as needed. Our profile HMM has a fixed size and topology once the parameter L is set during the initialization. A second difference is *early vs late merging*. Here, the progressive method merges information from features only at the distance level (computing the cost matrix C), which could be called late-merging. In contrast, the profile HMM learns a consensus in the form of a sequence of means for the match states: for a given match state, the mean is computed during the maximization step in Baum-Welch as a weighted sum of feature vectors (where the weights correspond to the posterior probabilities computed using the forward-backward procedure). Therefore, the averaging of information is already done at the feature level, which could be called early-merging. A third difference is the *distance measure*. In a progressive method one is free to choose or design a distance measure to compare feature vectors. In a profile HMM, distances correspond to observation probabilities and as such one typically has to choose from specific families of distributions (like the Gaussian family). While this is a limitation on the one hand, it enables the learning of parameters. In our case, we can learn and adapt the covariance matrices, which conceptually can be regarded as local feature distances adapted to the sequences. A fourth difference is the *greediness of updates*. To process a single sequence, the progressive method computes an alignment with the current template and updates the template before the next sequence is processed. The profile HMM employs the forward-backward procedure as part of Baum-Welch to compute the posterior, which conceptually can be interpreted as computing a soft alignment between each given feature sequence and the states in the profile HMM. Interpreted this way, in each iteration of Baum-Welch the profile HMM first computes an alignment for every single sequence before it updates its parameters. In this respect, the progressive method is more greedy compared to the profile HMM.

Overall, it is difficult to argue whether, for example, the increase in flexibility resulting from adaptability in size could give our progressive method an advantage over our profile HMM, or whether the greedy updates of the progressive method not only lower the computational costs but also reduce its alignment accuracy (as the profile HMM updates might be more robust due to taking all feature sequences into account). Therefore, we conduct in this section a series of experiments to assess the alignment quality of both methods under real-world conditions. To maximize the comparability, we use the same features for both methods and choose the best parameter configuration we could identify, i.e. the set of parameters maximizing the alignment accuracy, as described below. Furthermore, to identify whether our methods indeed have benefits over standard synchronization methods, we include the results for two widely-used pairwise methods [13], [14]. While the method presented in [14] uses a different set of features, the method presented in [13] is directly comparable to our methods as the same types of features are being used.

A. Dataset and Settings

1) Dataset: For our evaluation, we use a dataset consisting of 288 recordings for five of Chopin's Mazurkas, with 30-80 individual performances per piece, see Table I. The dataset is highly useful in our context for several reasons. First, interpretations of Mazurkas are often quite expressive leading to considerable differences in terms of timing, dynamics, balance, articulation and playing style. Second, the recordings were made in a time span ranging from 1931 to 2002 across a

TABLE I
CHOPIN MAZURKAS AND THEIR IDENTIFIERS USED IN OUR EXPERIMENTS. THE LAST TWO COLUMNS INDICATE THE NUMBER OF PERFORMANCES AVAILABLE FOR THE RESPECTIVE PIECE AND THE NUMBER OF EVALUATED UNIQUE PAIRS.

ID	Piece	No. Rec.	No. Pairs
M17-4	Opus 17 No. 4	62	1891
M24-2	Opus 24 No. 2	62	1891
M30-2	Opus 30 No. 2	34	561
M63-3	Opus 63 No. 3	81	3240
M68-3	Opus 68 No. 3	49	1176

TABLE II
ALIGNMENT ERROR (MEAN AND STANDARD DEVIATION OF AVERAGE BEAT DEVIATION, IN MILLISECONDS) FOR THREE TYPES OF ALIGNMENT METHODS AND A RANDOM BASELINE

	Pairwise I [14]		Pairwise II [13]		Profile HMM		Progressive Alignment		Baseline	
ID	mean	std	mean	std	mean	std	mean	std	mean	std
M17-4	116	638	68	19	62	12	59	12	3997	1908
M24-2	79	35	39	20	33	9	31	6	2726	2485
M30-2	69	121	30	8	32	7	30	5	2403	1401
M63-3	181	1332	46	32	39	11	40	11	2874	1846
M68-3	212	1444	58	23	51	19	46	13	1947	1177

wide range of venues, often resulting in extensive differences regarding the noise level, reverberation and room acoustics, acoustical properties of the instrument in use, recording equipment and audio quality as well as stylistic choices typical for a specific time period. Overall, these differences present substantial challenges to an automatic alignment method.

To evaluate our methods, we can exploit another unique property of the dataset. In particular, corresponding positions across different performances were manually annotated on the beat level as part of the Mazurka project[1], which enables a straightforward evaluation of automatic alignment methods as described next. Since handling structural differences is out of our scope, we exclude performances with structural differences (such as additional repetitions of a part of a piece) from our experiments.

2) Evaluation Measure: We use the manually annotated beat positions as follows to evaluate the alignment accuracy: given an alignment path between two versions of a piece, we locate for each annotated beat position in the one version the corresponding position in the other version. The absolute differences between the manually annotated beat positions and those obtained from the alignment are computed and averaged for all beats. The average (in milli-seconds) is employed as the evaluation measure, which is referred to as the *average beat deviation (ABD)* in the following. It is measured for each Mazurka and each pair of recordings in our experiments. Note that the number of pairs for one Mazurka is a binomial coefficient, for example, for M17-4 our setup contains 62 recordings, which results in $\binom{62}{2} = 1891$ unique pairs and corresponding ABD values, see Table I. Further, to increase the interpretability of the evaluation results, we include in Table II the results for a baseline method that simply linearly stretches the shorter to the longer recording to obtain an alignment.

3) Features and Parameters: For the pairwise method [13] and our two joint alignment methods, we use a combination of CENS [27], which is a type of chroma feature with uniform energy distribution, and DLNCO features [13], which estimate onset positions separately for each chroma, both with a 20ms temporal resolution. For the pairwise method and progressive alignment, we use the cosine distance for CENS and the Euclidean distance for DLNCO, as proposed in [13]. Further, we set the weights $(w_1, w_2, w_3) = (2, 1.5, 1.5)$ for both methods, use a gap penalty $\mathcal{C}_G = 3.6$ and sort the feature sequences to be aligned according to their length from short to long. (We investigate the influence of these parameters in more detail in Section V). The pairwise method described in [14] employs spectrogram-based features and the Euclidean distance to compare them. We use the default settings provided with the method.

[1]http://www.mazurka.org.uk

B. Comparison Between the Pairwise and Joint Alignments

Before we begin our more detailed investigation of individual components in our methods, we start with a more general comparison of the alignment accuracy of the pairwise and joint alignment methods. The distribution of the average beat deviation (ABD) values for all pairs is summarized for each of the five Mazurkas separately in Table II as well as in the boxplots[2], shown in Fig. 5. As a reference, we additionally include in Table II the results of another widely used pairwise method, referred to as method I [14], and the baseline method, which uses a linear stretch as discussed in Section IV-A2.

As shown in Table II, both joint alignment methods reduce the mean ABD compared to the pairwise method II [13], for most pieces. For example, the mean ABD for M68-3 drops from 58ms using pairwise alignment, to 51ms with the profile HMM alignment (decrease by 12%), and even lower to 46ms with the progressive alignment (decrease by 21%). On average, the mean ABD drops by 12% using the profile HMM and by 15% using progressive alignment. However, a more considerable improvement resulting from the joint alignment methods is a higher robustness. As can be seen from Fig. 5, the inter-quartile range is smaller for all five pieces using either of the two joint alignment methods, and the number of large-value outliers is drastically reduced. We can also measure this improvement by the decrease of the standard deviation (std), which for M68-3 is 17% using the profile HMM (drops from 23ms to 19ms) and 43% using progressive alignment (from 23ms to 13ms). This decrease is even greater for other Mazurkas (M24-2 and M63-3). On average, the standard deviation of ABD is reduced by 51% using the profile HMM and 58% using progressive alignment.

Overall, the two joint alignment methods are more stable compared to pairwise alignment, as both of them provide a higher robustness against large alignment errors, which also leads to an increase in alignment accuracy. As an exception, the improvement on M30-2 is limited, as the mean ABD using

[2]We use standard boxplots: the box gives the $25th$ and $75th$ percentiles (p_{25} and p_{75}), where the center bar indicates the median. The whiskers extend to the smallest data point greater than p_{25} - $1.5(p_{75}$ - $p_{25})$ and the largest data point less than p_{75} + $1.5(p_{75}$ - $p_{25})$, and the outliers are plotted as red crosses.

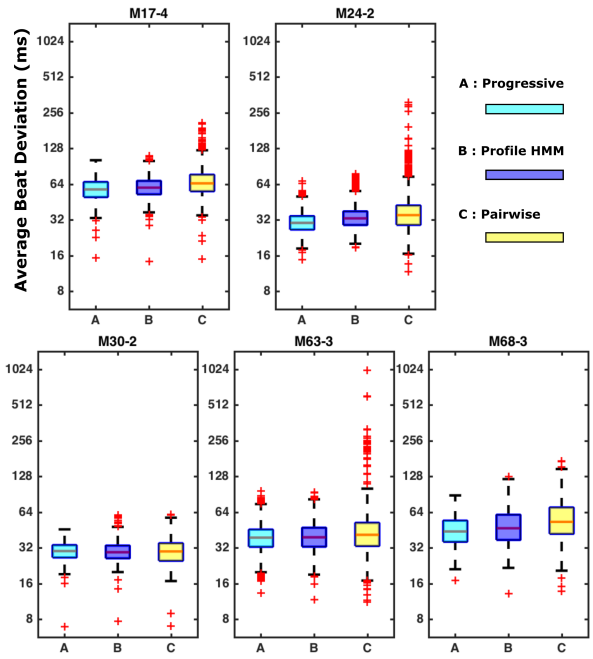

Fig. 5. Comparison of the pairwise alignment method [13] with our proposed progressive alignment method and profile HMM method. The boxplots illustrate the distribution of the average beat deviation values for each Mazurka separately on a logarithmic scale.

the progressive alignment is the same as using pairwise alignment, while the profile HMM is a bit worse, and the std drops only slightly (from 8ms to 7ms using profile HMM and to 5ms using progressive alignment). However, the experimental results indicate that this piece is relatively easy to align, since the mean ABD using pairwise alignment is 30ms (which is already low compared to the feature resolution level of 20ms) and the outliers are few and not as extreme as in other pieces. In this case, there is less room for the joint alignment methods to improve. This result matches the main effect we observe from the joint alignment, which is a gain in robustness against strongly incorrect alignments.

To test this hypothesis further, we conducted another experiment to show which error level is improved the most by our methods. To this end, we show in Fig. 6 a histogram of the deviation for all individual beats using all alignment pairs without averaging (corresponding to around 2.5 million evaluated beats). It shows that both joint alignment methods reduce the number of alignment errors clearly in the range of 100ms - 1000ms beat deviation.

We illustrate the superior robustness of our joint alignment methods over pairwise methods, with an example aligning performances of Op. 24 No. 2 by Luisada and Richter. Figure 1 shows an excerpt of the alignment which is problematic for pairwise method II [13]. As shown in the corresponding score, the six measures are mainly composed of repeated notes or chords with expressive markings. In addition to differences in balance (the relative loudness), as we can see from the CENS features, the two performers also play differently with regard to the timing. Furthermore, the two recordings contain different degrees of noise. In the presence of the above

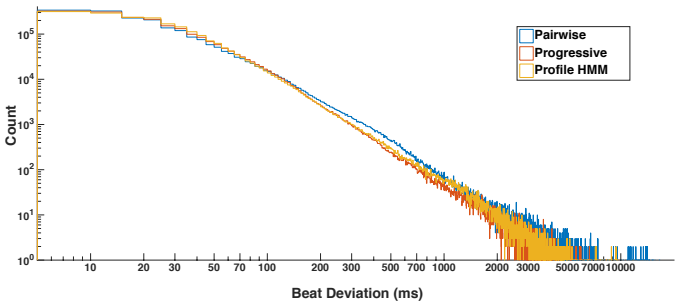

Fig. 6. Histograms of beat deviation using the pairwise alignment method, the progressive alignment and profile HMM method.

differences, the pairwise method fails to identify the correct alignment, see the solid path in Fig. 1, compared to the annotated beat positions (red dots). The other two alignments, which are shown as dashed and dotted paths in Fig. 1, result from our two joint alignment methods. In computing them, we include five other recordings, whose information helps to stabilise the alignment. As a consequence, we see that these two paths coincide almost always with the ground truth annotations.

C. Comparison of the Two Joint Alignment Methods

As shown in Table II and Fig. 5, the two joint alignment methods have a similar alignment accuracy and robustness, with the profile HMM having a slightly higher mean and std ABD for some pieces. To find out whether these small differences are statistically significant, we conducted a t-test to compare ABD values for all alignment pairs using the PA and PP methods. It indicates that there is a statistically significant difference in the ABD value using PA method ($M = 42, SD = 14$) and PP method ($M = 44, SD = 16$); $t(8758) = 20.4$, $p = 1e-90$. However, despite the significance, the difference between the two is relatively small in this experiment, which is also reflected by Cohen's measure for effect size: $d_s = 0.1$, indicates that the statistical significance is mainly reached due to the fairly large sample size. Therefore, in the next section, we will conduct a series of experiments on both joint alignment methods, to better understand their behaviour in other scenarios, to give an in-depth analysis of the influence of their parameters and to show possible extensions to further improve their performance.

V. Further Investigations of the Joint Alignment Method

In this section, we conduct six groups of additional experiments to further understand the behaviour of our joint alignment methods. We start with investigating the effect the number of available performances has on our methods. Next, we study the influence of the gap concept and the gap penalty parameter on the progressive alignment method, followed by an analysis of the influence of the order in which recordings are aligned. After that, we implement Viterbi training as an alternative model training method to the Baum-Welch process, in order to further accelerate the profile HMM.

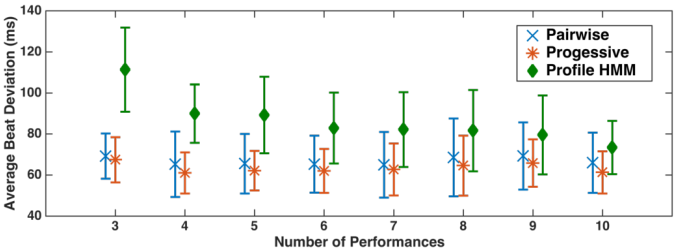

Fig. 7. Comparison between the pairwise alignment [13] and two joint alignment methods for subset Experiments.

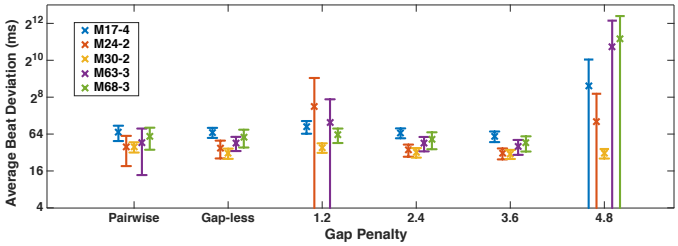

Fig. 8. Average beat deviation (ABD) values for five Mazurka pieces with progressive alignment using a gap-less variant and different values of gap penalty, compared with the pairwise alignment method [13]. The cross markers represent the mean ABD and the error bars show the standard deviation.

As a reverse idea, we also introduce an iterative extension to our progressive alignment method and study whether it can be used to exchange computation time for an increase in alignment robustness. Finally, we provide the evaluation results for new pieces with highly precise ground truth to further test how our methods behave under clean recording conditions (compared to the highly varied accoustic scenarios available in the Mazurka dataset).

A. Subset Experiments

In the previous experiments, we used large numbers (30 to 80) of performances of each piece to perform joint alignments. However, there is not always such a large number of different versions available for the same piece. Therefore, in this experiment, we investigate how many recordings we need to observe an improvement in robustness using the joint alignment methods compared to a pairwise method.

We perform experiments with subsets of different sizes ranging from 3 to 10 recordings. For each size, we randomly choose 10 sets from all the recordings of a given piece. Numerical results for the pairwise alignment method [13] are compared with both PA and PP methods in Fig. 7. As shown, the progressive alignment method decreases the mean and std ABD for subsets of all sizes steadily, compared to the pairwise method. The difference when there are only three recordings available is relatively small but still measurable, and it becomes more pronounced when more recordings are included in the alignment procedure. The results indicate that progressive alignment can improve the alignment accuracy and robustness even with a small set of recordings, i.e. it is not necessary to have a large number of versions in order to benefit from our method.

On the other hand, the Profile HMM method is worse in terms of mean and std ABD than both pairwise and progressive alignment methods when only a few recordings are available. The main reason here is that the profile HMM employs training data to adjust the internal sequence representation to the given data, and with so little training data this capability simply cannot yet unfold its advantages. Its performance improves with larger subsets, as more data is available in the model training. This behaviour could indicate that the increase in alignment accuracy for profile HMM based methods as reported in some bio-informatics-related publications might only be achievable if a similar number of training sequences is available. Since there are often several thousand sequences available in bio-informatics [58], the situation is quite different to music processing where such a high number cannot be expected.

B. Gap Penalty

To avoid a possible temporal uncertainty caused by copying features, we insert a special gap symbol when updating the template (Section III-A). We study the influence of these gaps on the alignment accuracy by experimenting with different values of the gap penalty parameter and a gap-less variant.

For the gap-less variant we use the simple strategy described in Section III-A and set $\widetilde{z}_\ell = (z^1_{n_\ell}, \ldots, z^{k-1}_{n_\ell}, x^k_{m_\ell})$. Comparing the results for this gap-less variant with the baseline pairwise method in Fig. 8, we can see that the gap-less version leads to small improvements, mostly with respect to robustness as indicated by the decrease in dispersion. However, these improvements are more pronounced using the proposed progressive method with a gap penalty value of 3.6. Further, the gap-less variant does not reduce the mean ABD compared to the pairwise alignment. This behaviour could indicate that copying the features to stretch the newly aligned sequence, as done in the gap-less variant, indeed leads to a temporal uncertainty in the features causing the loss of alignment accuracy compared to the gap-variant.

However, from Fig. 8 we can also see that the value of the gap penalty needs to be sufficiently large, at least larger than the maximum value of the local cost measure (which is 3.0 in our case), to ensure every gap is sufficiently penalised. On the other hand, if the value is too large, features in the new sequence $x^k_{m_\ell}$ are not likely to get aligned to the z_l if it contains a gap which can lead to a loss of accuracy as well. We found 3.6 a suitable value for the gap penalty during the development of the method using only M17-4. As seen in Fig. 8, this value yields the best results for the remaining Mazurkas as well. Furthermore, it works well with additional pieces in Section V-F.

C. Alignment Order

As described in Section III-A, the template Z in our progressive method is built up gradually by successively aligning the feature sequences $X^1, \ldots, X^K$. The order in which they are aligned should be chosen with care for two reasons. Firstly, feature sequences at the beginning have less information from other versions to stabilise the alignment. Secondly, errors made

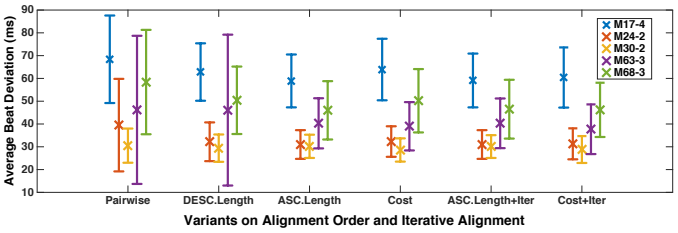

Fig. 9. Average beat deviation(ABD) values for five Mazurka pieces with progressive alignment using different alignment orders and the iterative extension, compared with the pairwise alignment method [13].

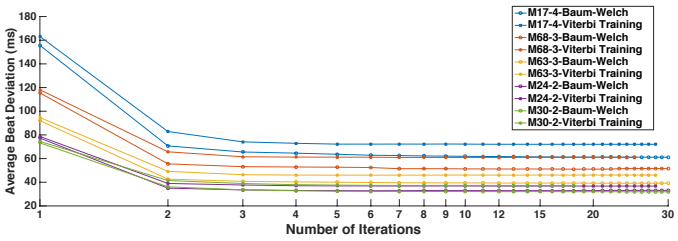

Fig. 10. The convergence of average beat deviation with increasing number of iterations for two model learning methods.

at an early stage may propagate to the following alignments. Therefore, we compare four different ordering strategies in our next experiments: The first strategy is ordering versions randomly. Next, we use two length-based ordering strategies, where versions are sorted by their duration in ascending or descending order. For the last strategy, we try to find an order for the sequences such that each sequence being aligned is the easiest to be aligned among all remaining sequences, in some sense. More precisely, we first compute for each pair of recordings an alignment and a corresponding total cost using the baseline pairwise method [13]. We normalize each cost by dividing it by the length of the corresponding alignment path. The pair with the smallest normalized cost defines the first two feature sequences to be aligned, i.e. X^1 and X^2. Next, we set X^3 to the feature sequence where the sum of its normalized costs to X^1 and X^2 is the smallest among all remaining sequences. We continue choosing the next version that has the lowest sum of normalized costs between itself and each one of the previously placed versions. This procedure is repeated until all recordings are sorted. Note that this strategy is considerably more computationally expensive than the first three.

Results are shown in Fig. 9, where we excluded the random order strategy as the resulting error was relatively high and would have occluded the nuances in the other strategies. As indicated by the results, the alignment order is indeed important in progressive alignment. The progressive alignment with a descending length-based order shows improvements in both accuracy and robustness over the pairwise method for most pieces. The ascending length-based order leads to an even better result. The possible reason could be that the template monotonically grows in length with each sequence being aligned: with a descending length based order, the difference in length between the template and the sequence to be aligned will become large when aligning the last several sequences, much larger than for the ascending length-based order where the template length grows slowly with the sequences being aligned. That may lead to a slight alignment accuracy drop when aligning shorter sequences at the end as the DTW weights in use have a slight bias in favour of the main diagonal direction, i.e. $(w_1 < w_2 + w_3)$ (Section IV-A). Both the ascending length-based and the cost-based order strategy decrease the mean ABD and the standard deviation without any significant differences between them. Due to the considerable difference in computational costs between these two strategies, we propose the use of the ascending length-based order.

D. Viterbi Training

Since the progressive alignment and the profile HMM have comparable alignment accuracies on larger datasets (Section IV), we compare their computational complexities to see whether other factors contribute to choosing one approach over the other when many versions of a piece are available. Therefore, we now inspect the computational complexity of both methods. To this end, let K be the number of recordings, each having about N features. To align the k-th recording to the template, the progressive alignment method computes $k-1$ cost matrices each with a time and memory requirement of $O(N^2)$ (the acceleration technique described in III.C with a fixed number of feature resolutions does not change the complexity level for this step). Since we repeat this step K times, we compute $\frac{K(K-1)}{2}$ cost matrices, thus the method is in $O(K^2N^2)$ (just as standard pairwise methods). In a single Baum-Welch iteration of the profile HMM, we compute K forward matrices and K backward matrices of size $3N^2$ (since we have three states per feature in the profile HMM). If we set the number of Baum-Welch iterations to a fixed value independent of the number of available recordings, the overall complexity is in $O(KN^2)$. Therefore, for a high number of recordings, the profile will eventually be the preferable approach, as the complexity is lower and with a high number of recordings the difference in alignment accuracy between the PA and PP method vanishes as well. In practice, with 10 Baum-Welch iterations, the runtime for the profile HMM will be lower for more than ≈ 120 recordings, as in this case K becomes higher than the ratio of constant factors influencing the absolute runtime of the profile HMM to that of progressive alignment (assuming similar runtime costs for the observation probabilities and the local cost measure).

The number of iterations we need, however, depends on the convergence behaviour of the method. Therefore, we conducted an experiment to investigate this behaviour. More precisely, Fig. 10 shows the average beat deviation for each piece after each Baum-Welch iteration. As we can see, the method typically converges rather quickly, with only little change after the first five to ten iterations (which motivated us to limit the number of iterations to 10 in the initial experiment).

A further technique often used in large scale procedures in speech processing to accelerate the training is *Viterbi Training* [24]. Here the idea is to replace the forward-backward

procedure with a simple Viterbi decoding. This way, instead of a soft value encoding the probability of being in a certain state at a certain time frame, the Viterbi decoding makes a hard choice and sets the probability of the state-time pair to 1 if it is on the most probable path, and to 0 otherwise. The parameter update remains conceptually identical. In practice, Viterbi training often converges faster than Baum-Welch in terms of the number of iterations but is more prone to local minima of the likelihood function due to the hard decision made during the decoding. Further, a single iteration with Viterbi training is about twice as fast as one iteration of Baum-Welch as the backward procedure becomes unnecessary.

We implemented Viterbi training for our profile HMM and include the convergence results in Fig. 10. As we can see, while each iteration is indeed more efficient with Viterbi training, the number of iterations necessary to reach convergence is about the same. Further, with Viterbi training, there is a slight but consistent loss of alignment accuracy caused by not maximizing the likelihood function as Baum-Welch does [24], which makes Viterbi training more likely to get stuck in a local optimum. Therefore, Viterbi Training could be most useful as an alternative to Baum-Welch process, to accelerate the model training if the number of available recordings is very high, however, at the cost of a slight drop in alignment accuracy.

E. Iterative Alignment

As mentioned in Section IV, progressive alignment is greedier regarding the updating process. Intuitively, this greediness may lead to an accuracy drop, as reported in some bioinformatics tasks. In particular, the first alignments need to be more reliable as they have less information available and, at the same time, will influence the alignment with all remaining sequences. To circumvent this potential problem, we now introduce an iterative extension to our progressive alignment that can be used to further refine the template. The basic idea is to remove individual versions from the template and realign them to the remaining template. Specifically, we take out one version at a time, starting from the first one, and perform the alignment between this version and the template of the remaining versions. We evaluate the resulting template after re-alignment using a score value, defined as the sum of the alignment costs between all pairs in this template (which can easily be extracted from the template). If the alignment score decreases, we keep the updated template, otherwise, we restore the previous template. This process of re-alignment is continued until no further improvement can be achieved.

We test the iterative refinement process with both ascending length based order and cost based order. As shown in Fig. 9, the iterative process does not lead to a substantial improvement for any of the five Mazurka pieces. Overall, in our experiments, we found that our progressive alignment is able to deliver alignments of both high accuracy and robustness with a single pass using a suitable alignment order.

F. Further evaluation

Although the Mazurka data is highly varied in terms of acoustic conditions and expressive local tempo variations (Section IV-A), the pieces are all of the same style and by one composer. Therefore, we now conduct on additional experiment using a set of four excerpts compiled in [59] from: Mozart Piano Sonata No. 11 in A major, KV331 first movement, Schubert German Dance D.783, No. 15, Chopin Etude in E major, Op. 10, No.3, and Chopin Ballade in F major, Op. 38. Each excerpt has 22 performances by skilled pianists recorded on a Bösendorfer computer-monitored piano. Compared to the Mazurka dataset, there are several major differences. First, all recordings were made using the same instrument under the same recording conditions and at the same time, such that the acoustic conditions do not differ much within the dataset. Second, the recording quality is very high and contains only little reverberation. Third, compared to the Mazurka pieces with manually annotated beat positions, this dataset contains precise onset annotations for each note. To account for the higher quality annotations, we change the evaluation measure from average beat deviation (ABD) to average note onset deviation in this section.

TABLE III
COMPARING THE PAIRWISE ALIGNMENT, PROFILE HMM AND PROGRESSIVE ALIGNMENT METHOD IN TERMS OF AVERAGE NOTE ONSET DEVIATION (IN MILLISECONDS)

	Pairwise II [13]		**Progessive**		**Profile HMM**	
Piece	mean	std	mean	std	mean	std
KV331	21	4	21	4	20	4
D783	27	7	24	4	27	8
Etude	26	6	24	3	24	3
Ballade	30	8	29	5	31	8

By providing cleaner acoustic conditions, we can use this dataset to test whether our methods also improve the alignment accuracy in less difficult scenarios, or whether a pairwise method can translate the clean conditions into higher accuracies than our proposed methods. The results for the pairwise alignment [13] and the two joint alignment methods are shown in Table III, where we used the same settings as described in Section IV-A3. First, we can see that the results reflect the recording quality in this dataset, with relatively low alignment errors for all three methods. Further, we can see that also using this dataset our joint alignment methods slightly improve the mean of the alignment error, with the progressive alignment slightly ahead of the profile HMM. More importantly, we observe a similar behaviour regarding the robustness of the alignments as before, with a considerably lower standard deviation for the joint methods: compared to the pairwise method, our progressive alignment again lowers the standard deviation by between 38% and 50% – despite the higher audio quality. These results demonstrate that our method indeed can be used to remove many outlier alignments, where the pairwise method fails to compute an accurate alignment.

VI. CONCLUSION

In this paper, we introduced two methods for the joint alignment of multiple performances of a piece of music: a progressive alignment (PA) and a probabilistic profile (PP) method. As demonstrated by our experiments using recordings

of Chopin Mazurkas, both methods can be used to improve the alignment accuracy and robustness over state-of-the-art pairwise methods. An increase in accuracy using a method from the PP family over a member of the PA family as reported in bioinformatics could not be observed in our music synchronization scenario, but the superior computational complexity of our PP method makes it an interesting option if the number of available recordings is a hundred or higher. We conducted additional experiments to investigate the behaviour of our joint alignment methods by testing the influence of various parameters and analyzed the performance of various extensions aiming to increase their alignment accuracy and computational efficiency. In particular, experiments with smaller datasets showed that our method can outperform state-of-the-art pairwise methods even if only a small set of recordings is available.

ACKNOWLEDGMENT

This work was partly funded by the China Scholarship Council (CSC) and EPSRC Grant EP/J010375/1 and EP/L019981/1.

REFERENCES

[1] S. Wang, S. Ewert, and S. Dixon, "Robust joint alignment of multiple versions of a piece of music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 83–88.

[2] A. Arzt, S. Böck, S. Flossmann, H. Frostel, M. Gasser, and G. Widmer, "The complete classical music companion v0.9," in *Proceedings of the AES International Conference on Semantic Audio*, London, UK, 18–20 2014, pp. 133–137.

[3] N. Montecchio and A. Cont, "A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 193–196.

[4] M. Müller, M. Clausen, V. Konz, S. Ewert, and C. Fremerey, "A multimodal way of experiencing and exploring music," *Interdisciplinary Science Reviews (ISR)*, vol. 35, no. 2, pp. 138–153, 2010.

[5] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, pp. 1138–1151, 2008.

[6] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[7] S. Ewert, B. Pardo, M. Müller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, May 2014.

[8] N. Hu, R. B. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[9] N. Orio and F. Déchelle, "Score following using spectral analysis and Hidden Markov Models," in *Proceedings of the International Computer Music Conference (ICMC)*, 2001, pp. 125–129.

[10] C. Joder, S. Essid, and G. Richard, "A conditional random field framework for robust and scalable audio-to-score matching," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2385–2397, 2011.

[11] C. Raphael, "A hybrid graphical model for aligning polyphonic audio with musical scores," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004, pp. 387–394.

[12] Z. Duan and B. Pardo, "A state space model for online polyphonic audio-score alignment," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 197–200.

[13] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.

[14] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, London, GB, 2005, pp. 492–497.

[15] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, "In search of the Horowitz factor," *AI Magazine*, vol. 24, no. 3, pp. 111–130, 2003.

[16] C. S. Sapp, "Comparative analysis of multiple musical performances," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007, pp. 497–500.

[17] M. Müller, V. Konz, A. Scharfstein, S. Ewert, and M. Clausen, "Towards automated extraction of tempo parameters from expressive music recordings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, 2009, pp. 69–74.

[18] C. C. Liem and A. Hanjalic, "Expressive timing from cross-performance and audio-based alignment patterns: An extended case study," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, 2011, pp. 519–524.

[19] N. Montecchio and A. Cont, "Accelerating the mixing phase in studio recording productions by automatic audio alignment," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, United States, 2011.

[20] D. Baaran, A. T. Cemgil, and E. Anarm, "A probabilistic model-based approach for aligning multiple audio sequences," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 23, no. 7, pp. 1160–1171, 2015.

[21] C. Raffel and D. P. W. Ellis, "Large-scale content-based matching of MIDI and audio files," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[22] G. H. Wakefield, "Mathematical representation of joint time-chroma distributions," in *Proceedings of the SPIE International Symposium on Optical Science, Engineering, and Instrumentation*, 1999, pp. 637–645.

[23] E. Gómez, "Tonal description of music audio signals," Ph.D. dissertation, UPF Barcelona, 2006.

[24] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. New York, USA: Cambridge University Press, 1999.

[25] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press, 1997.

[26] F. S.-M. Pais, P. de Cássia Ruy, G. Oliveira, and R. S. Coimbra, "Assessing the efficiency of multiple sequence alignment programs," *Algorithms for Molecular Biology*, vol. 9, no. 1, pp. 1–8, 2014.

[27] M. Müller, F. Kurth, and M. Clausen, "Audio matching via chroma-based statistical features," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 288–295.

[28] R. B. Dannenberg, "An on-line algorithm for real-time accompaniment," in *Proceedings of the International Computer Music Conference (ICMC)*, 1984, pp. 193–198.

[29] B. Vercoe, "The synthetic performer in the context of live performance," in *Proc. International Computer Music Conference (ICMC)*, 1984, pp. 199–200.

[30] N. Orio and D. Schwarz, "Alignment of monophonic and polyphonic music to a score," in *Proceedings of the 2001 International Computer Music Conference*, 2001, pp. 155–158.

[31] R. J. Turetsky and D. P. Ellis, "Ground-truth transcriptions of real music from force-aligned MIDI syntheses," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Baltimore, USA, 2003, pp. 135–141.

[32] A. Cont, "A coupled duration-focused architecture for real-time music-to-score alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 974–987, 2010.

[33] R. B. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proceedings of the International Computer Music Conference (ICMC)*, San Francisco, USA, 2003, pp. 27–34.

[34] A. Cont, "Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical HMMs," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, Toulouse, France, 2006.

[35] B. Niedermayer, "Towards audio to score alignment in the symbolic domain," in *Proceedings of the Sound and Music Computing Conference (SMC)*, Porto, Portugal, 2009, pp. 77–82.

[36] C. Joder, S. Essid, and G. Richard, "Learning optimal features for polyphonic audio-to-score alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2118–2128, 2013.
[37] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.
[38] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.
[39] C. Raphael, "Automatic segmentation of acoustic musical signals using Hidden Markov Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 360–370, 1998.
[40] R. Miotto, N. Montecchio, and N. Orio, "Statistical music modeling aimed at identification and alignment," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 187–212.
[41] A. Maezawa, K. Itoyama, Y. Kazuyoshi, and H. G. Okuno, "Bayesian audio alignment based on a unified model of music composition and performance," *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 233–238, 2014.
[42] M. Müller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 192–197.
[43] S. Salvador and P. Chan, "FastDTW: Toward accurate dynamic time warping in linear time and space," in *Proceedings of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.
[44] R. Macrae and S. Dixon, "Accurate real-time windowed time warping," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, Netherlands, 2010, pp. 423–428.
[45] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.
[46] A. Maezawa, K. Itoyama, K. Yoshii, and H. G. Okuno, "Unified inter- and intra-recording duration model for multiple music audio alignment," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2015 IEEE Workshop on*, Oct 2015, pp. 1–5.
[47] A. Arzt and G. Widmer, "Real-time music tracking using multiple performances as a reference," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain, 2015.
[48] M. G. Bergomi, "Dynamical and topological tools for (modern) music analysis," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2015.
[49] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, "Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrfs," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 993–1000.
[50] K. Katoh and D. M. Standley, "MAFFT multiple sequence alignment software version 7: improvements in performance and usability," *Molecular biology and evolution*, vol. 30, no. 4, pp. 772–780, 2013.
[51] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega," *Molecular Systems Biology*, vol. 7, no. 1, 2011.
[52] G. ten Holt, M. Reinders, and E. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Proceedings of the Advanced School for Computing and Imaging*, 2007.
[53] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll, "A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams," *Neurocomputing*, vol. 73, pp. 366–380, 2009.
[54] S. Wang, S. Ewert, and S. Dixon, "Compensating for asynchronies between musical voices in score-performance alignment," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, 2015, pp. 589–593.
[55] R. B. Dannenberg and C. Raphael, "Music score alignment and computer accompaniment," *Communications of the ACM, Special Issue: Music Information Retrieval*, vol. 49, no. 8, pp. 38–43, 2006.
[56] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
[57] S. J. Cox, "Hidden Markov models for automatic speech recognition: Theory and application," in *Speech and Language Processing*, C. Wheddon and R. Linggard, Eds. London, UK: Chapman & Hall, 1990, pp. 209–230.
[58] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "Balibase 3.0: latest developments of the multiple sequence alignment benchmark," *Proteins: Structure, Function, and Bioinformatics*, vol. 61, no. 1, pp. 127–136, 2005.
[59] W. Goebl, "Numerisch-klassifikatorische Interpretationsanalyse mit dem 'Bösendorfer Computerflügel'," Master's thesis, Universität Wien, 1999.

Siying Wang received the BSc(Eng) degree in telecommunication engineering from Beijing University of Posts and Telecommunications in 2009. She is currently pursuing her doctoral degree at the Centre for Digital Music, Queen Mary University of London (United Kingdom). Her research interests include audio signal processing, music information retrieval and musical performance study.

Sebastian Ewert received the M.Sc./Diplom and Ph.D. degrees (summa cum laude) in computer science from the University of Bonn (svd. Max-Planck-Institute for Informatics), Germany, in 2007 and 2012, respectively. After a postdoc at the Centre for Digital Music, Queen Mary University of London (United Kingdom), he became lecturer for signal processing in the centre in 2015. Currently, he is additionally holding a research position in the EPSRC programme Fusing Audio and Semantic Technologies (FAST) and is one of the leaders of the Machine Listening Lab.

Simon Dixon is a Reader (Assoc. Prof.), Director of Graduate Studies and Deputy Director of the Centre for Digital Music at Queen Mary University of London. He has a PhD in Computer Science (Sydney) and LMusA diploma in Classical Guitar. His research interests include high-level music signal analysis, computational modelling of musical knowledge, and the study of musical performance. Particular areas of focus include automatic music transcription, beat tracking, audio alignment and analysis of intonation and temperament. He was President (2014-15) of the International Society for Music Information Retrieval (ISMIR), is member of the Editorial Board of the Journal of New Music Research (since 2011), and has published over 150 refereed papers in the area of music informatics.