A STATE SPACE MODEL FOR ONLINE POLYPHONIC AUDIO-SCORE ALIGNMENT

Zhiyao Duan and Bryan Pardo

Northwestern University
Department of Electrical Engineering and Computer Science
2133 Sheridan Road, Evanston, IL 60208, USA.
zhiyaoduan00@gmail.com, pardo@northwestern.edu

ABSTRACT

We present a novel online audio-score alignment approach for multi-instrument polyphonic music. This approach uses a 2-dimensional state vector to model the underlying score position and tempo of each time frame of the audio performance. The process model is defined by dynamic equations to transition between states. Two representations of the observed audio frame are proposed, resulting in two observation models: a multi-pitch-based and a chroma-based. Particle filtering is used to infer the hidden states from observations. Experiments on 150 music pieces with polyphony from one to four show the proposed approach outperforms an existing offline global string alignment-based score alignment approach. Results also show that the multi-pitch-based observation model works better than the chroma-based one.

Index Terms— Score following, audio-score alignment, online algorithm, realtime, hidden Markov model

1. INTRODUCTION

Score alignment is the task of aligning a musical performance with its corresponding score. It has been a research topic at least since 1984 [1, 2], and is still an active research topic. Typical applications of score alignment include accompanying a monophonic performance [3] and synchronizing multiple sources (video, audio, score, etc.) of music in a digital library [4].

Score alignment can be addressed *offline* or *online*. An offline algorithm can use the whole performance of a music piece. The online version (also called score following) cannot "look ahead" at future performance events when aligning the current event to the score.

In score alignment, although the musical score is usually a machine-readable MIDI file, the musical performance can be encoded as either MIDI or audio. We focus here on *audio-MIDI* alignment where the performance is audio and the score is encoded as MIDI. Performances can be classified as *monophonic* and *polyphonic*. Monophonic audio can be reliably transcribed by mature single-pitch detection techniques. Polyphonic pitch estimation remains an open area of research.

In this paper, we consider the most difficult case, i.e. online polyphonic audio-MIDI alignment. Reliable online polyphonic audio-midi alignment would support applications including automatic accompaniment of live music, real-time audio pitch correction, and real-time score-informed source separation.

Score following research began with following MIDI performances [1, 2, 5]. To follow monophonic audio performances, Puckette [6], Grubb and Dannenberg [7] proposed systems to follow singing voices. Orio and Dechelle [8] proposed a Hidden Markov Model (HMM)-based approach to follow different instruments and voices. Raphael [3] used a Bayesian network to follow and accompany a solo instrument.

For polyphonic audio, Grubb and Dannenberg [9] adopted string matching to follow a musical ensemble, where each instrument needs to be recorded by a close microphone and streamed into a monophonic pitch sequence. This method will not work where the instruments are not separately recorded. Ewert et al. [10] build a high resolution polyphonic audio-score alignment system using Dynamic Time Warping (DTW) with chroma onset features, which works in an offline fashion. Dixon's [11] online DTW algorithm follows piano performances, where each audio frame is represented by a onset-informed low-level feature vector. This, however, may have difficulties for instruments with smooth onsets like strings and winds. Cont [12] proposed a hierarchical HMM approach with Nonnegative Matrix Factorization (NMF) to follow piano performances. A spectral basis is learned for each pitch of the piano before-hand. Tuning issues make it difficult to create a basis set to cover arbitrary pitch-instrument combinations. In [13], a probabilistic inference framework with two coupled audio and tempo agents follows a polyphonic performance and estimate its tempo. This works well on single-instrument polyphonic audio. For multi-instrument polyphonic audio, more experiments are needed to evaluate performance.

We now describe a novel online score follower that can follow multi-instrument polyphonic audio performances, without requiring training on prior performances of the piece or on the instruments to be followed. Statistical results on 150 real music performances show the validity of the method.

2. METHOD

We decompose an audio performance into time frames and use a vector in a 2-dimensional state space to represent the underlying score position and tempo of each frame. As illustrated in Figure 1, for the n-th frame the hidden state vector is $\mathbf{s}_n = (x_n, v_n)^T$, where x_n is its score position (in beats), v_n is its tempo (in Beats Per Minute (BPM)) and T denotes matrix transposition. The frame is also associated with an observation, which is a vector of PCM encoded audio, $\mathbf{y}_n$. Our aim is to infer the current score position x_n from current and previous observations $\mathbf{y}_1, \cdots, \mathbf{y}_n$.

Here, x_n is drawn from the interval containing all score positions from the beginning to the end. v_n is drawn from the interval of all possible tempi $[v^l, v^h]$. In this work, we set the lowest tempo v^l to half of the notated score tempo and the highest tempo v^h to twice the score tempo. It is noted that the state space is continuous instead of discrete.

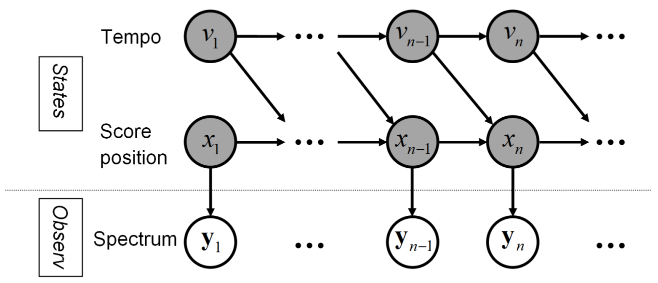

Fig. 1. Illustration of the state space model. Each column corresponds to a time-frame of the audio performance.

In the following sections, we define a process model to describe how the states transition, propose an observation model to describe the likelihood of generating an observation given a state, and find a way to infer the hidden states.

2.1. Process model

We use two dynamic equations to transition between the previous and current state. To update the score position, we use

$$x_n = x_{n-1} + l \cdot v_{n-1} \tag{1}$$

where l is the audio frame hop size. Thus, score position of the current audio frame is determined by the score position of the previous frame and the current tempo. To update the tempo, we use

$$v_n = \begin{cases} v_{n-1} + n_v & \text{if } z_k \in [x_{n-1}, x_n] \text{ for some } k \\ v_{n-1} & \text{otherwise} \end{cases} \tag{2}$$

where $n_v \sim \mathcal{N}(0, \sigma_v^2)$ is a Gaussian noise variable; z_k is the k-th note onset/offset time in the score. If the current score position has just passed a note onset or offset, the current tempo makes a random walk around the previous tempo according to a Gaussian distribution; otherwise the current tempo remains the same. The noise n_v is to account for possible tempo changes of the performer. Since a tempo change can only be perceived near an onset or offset, it is only introduced in the first line. Here, the standard deviation of n_v is set to a quarter of the notated score tempo. In addition, we can see that randomness is only introduced in tempo instead of score position. In this way, the score position estimates change more smoothly.

2.2. Observation model

The observation model presents the likelihood of observing an audio frame given a state, i.e. $p(\mathbf{y}_n|\mathbf{s}_n)$. Different audio representations (features) have been exploited in existing audio-score alignment systems [14]. In this paper, we use two kinds of representations, resulting two observation models that we compare: one based on multi-pitch information, and one based on chroma.

2.2.1. Multi-pitch-based model

Multi-pitch information is very useful to represent an audio frame in score following, since pitch information can be directly aligned to score information [12]. In this paper, we use the multi-pitch likelihood function which is defined in our previous work on multi-pitch estimation [15] to define the observation model $p(\mathbf{y}_n|\mathbf{s}_n)$,

$$p(\mathbf{y}_n|\mathbf{s}_n) = -\frac{C}{\ln(p(\mathbf{y}_n|\boldsymbol{\theta}))} \tag{3}$$

where C is the normalization factor for probability; $\boldsymbol{\theta}$ is the set of pitches indicated by the score at position x_n; and $p(\mathbf{y}_n|\boldsymbol{\theta})$ is the multi-pitch likelihood function defined in [15].

We calculate $p(\mathbf{y}_n|\boldsymbol{\theta})$ as described in [15]. Basically, a power spectrum is calculated from the audio frame $\mathbf{y}_n$. Peak regions and non-peak regions are identified from this spectrum and the multi-pitch likelihood function $p(\mathbf{y}_n|\boldsymbol{\theta})$ is defined separately in these two regions. Model parameters of $p(\mathbf{y}_n|\boldsymbol{\theta})$ are trained on isolated musical chords generated by 16 kinds of instruments with a wide pitch and dynamic range. This model performs well in estimating polyphonic pitches on multi-instrument polyphonic music pieces in [15]. Therefore, we believe it is a good choice to define the observation model for our score follower.

Note that we do not use $p(\mathbf{y}_n|\boldsymbol{\theta})$ as $p(\mathbf{y}_n|\mathbf{s}_n)$ directly, because it turns out that $p(\mathbf{y}_n|\boldsymbol{\theta})$ differs too much (usually tens of order of magnitude) for different pitches at different score positions. Since the probability calculated from the process model does not have this large difference for different score positions, the posterior probability of state will be strongly influenced by the observation model, while the process model will be almost ignored. If the observation model does not function well in some frame, the estimated score position may jump to an unreasonable position, although the process model tend to proceed from previous score position smoothly.

Also note that in evaluating an observation likelihood $p(\mathbf{y}_n|\mathbf{s}_n)$, we do not need to estimate the pitches in the audio frame. This is different from [12], where pitches of the audio frame are first estimated, then the observation likelihood is calculated based on the differences between the estimated pitches and the score-indicated pitches. By skipping the pitch estimation step, we reduce model risks caused by pitch estimation errors.

2.2.2. Chroma-based observation model

The chromagram is a good way to represent the harmonic content of an audio or score frame. It has been used in a number of offline score alignment approaches, e.g. [16]. In our work, chroma are extracted for both audio and MIDI. It is a 12-d vector, where each dimension corresponds to a pitch class. The amplitudes of all spectral bins of an audio frame that share a pitch class are summed to get the value of the pitch class in the audio chroma vector. For a score position, the dimension of the MIDI chroma vector is set to 1 if there is a score pitch in this pitch class; otherwise, it is set to 0.

Given the audio chroma vector $\mathbf{c}_a$ of the n-th audio frame and the score chroma vector $\mathbf{c}_m$ at score position x_n, we calculate the angle between the two vectors,

$$\alpha = \arccos\left(\frac{\mathbf{c}_a^T \mathbf{c}_m}{\|\mathbf{c}_a\|\|\mathbf{c}_m\|}\right) \tag{4}$$

where T denotes the vector transpose and $\|\cdot\|$ denotes the Euclidean norm. For the special case that only one vector is zero, i.e. either audio or score is silent but not both, α is defined as $\frac{\pi}{2}$. If both vectors are zero, then α is 0. Note that the range of α is $[0, \frac{\pi}{2}]$, since both vectors are nonnegative.

We assume that α has a Gaussian distribution centered at 0 with a standard deviation of 1. Then $p(\mathbf{y}_n|\mathbf{s}_n)$ is defined as

$$p(\mathbf{y}_n|\mathbf{s}_n) = C'p(\alpha) = C'\frac{1}{\sqrt{2\pi}}\exp\left\{-\alpha^2\right\} \tag{5}$$

where C' is the normalization factor to make $p(\mathbf{y}_n|\mathbf{s}_n)$ a probability distribution. It is noted that although α assumes a Gaussian distribution, $\mathbf{y}_n$, i.e. $\mathbf{c}_a$ in this model does not.

We use cosine angle distance instead of Euclidean distance to define the observation likelihood in Eq. (5) to make it loudness insensitive. This is because the loudness of the audio may vary from the loudness indicated by the score differently in different music performances.

2.3. Inference

Given the process model and the observation model, we want to infer the hidden state $\mathbf{s}_n$ of the current frame from current and past observations $\mathbf{Y}_{1:n} = (\mathbf{y}_1, \cdots, \mathbf{y}_n)$. From a Bayesian point of view, this means that we first estimate the posterior probability $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$, then decide its value using some criterion like maximum a posterior, mean or median. By Bayes' rule, we have

$$\begin{aligned} & p(\mathbf{s}_n|\mathbf{Y}_{1:n}) \\ = \; & C_n p(\mathbf{y}_n|\mathbf{s}_n) \int p(\mathbf{s}_n|\mathbf{s}_{n-1}) p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})\, d\mathbf{s}_{n-1} \quad (6) \end{aligned}$$

where C_n is the normalization factor; $\mathbf{s}_{n-1}$ is integrated over the whole state space; $p(\mathbf{y}_n|\mathbf{s}_n)$ is the observation model and $p(\mathbf{s}_n|\mathbf{s}_{n-1})$ is the process model.

In Eq. (6), we can see that $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$ is recursively updated from the posterior probability in the previous frame $p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})$. Therefore, if we initialize $p(\mathbf{s}_1|\mathbf{y}_1)$ in the first frame and keep updating it using Eq. (6) as each frame is processed, the inference can be done online.

This is the general formulation of online filtering (tracking). Particle filtering [17] is a way to solve the problem when the process model or the observation model is non-Gaussian, as are our observation models in Eq.(3) and (5). In particle filtering, the posterior distribution of state is represented and updated by a fixed number of particles together with their weights. We use the *bootstrap filter* [17], a particle filter variant that gives all particles the same weight in each iteration.

Prior to processing the first frame of audio, 1,000 particles are initialized to have score positions equal to the first beat and tempi assume a uniform distribution in the possible tempo range $[v^l, v^h]$. All particles have the same weight. When the n-th iteration starts, particles represent the posterior distribution $p(\mathbf{s}_{n-1}|\mathbf{Y}_{1:n-1})$ of $\mathbf{s}_{n-1}$. These particles are then updated according to state transition equations Eq. (1) and (2) and after this they represent the conditional distribution $p(\mathbf{s}_n|\mathbf{Y}_{1:n-1})$ of $\mathbf{s}_n$. Finally, the observation likelihood $p(\mathbf{y}_n|\mathbf{s}_n)$ is calculated for each particle according to Eq. (3) or Eq. (5) as its new weight.

The weights are then normalized and form a discrete distribution. The particles are sampled with replacement according to this distribution, then perturbed with a small Gaussian noise with zero mean and standard deviation of 0.01 beat for position and 1 BPM for tempo. After this, the newly sampled particles now represent the posterior distribution $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$ of $\mathbf{s}_n$. Finally, the mean of these particles is output as the estimated score position and tempo of the current frame.

In updating the tempo of each particle in Eq. (2), instead of using its previous tempo as v_{n-1}, we use the estimated tempo, i.e. the average tempo of all particles in the previous frame. This practical choice avoids that the particles become too diverse after a number of iterations due to the accumulation of randomness of n_v.

A common problem of particle filtering is degeneracy [17], where most particles have negligible weights after a few iterations. The bootstrap filter we use here alleviates this problem, since the resampling step in each iteration removes particles with small weights and maintains equal weights for all remaining particles. Also, perturbation after resampling assures the diversity of the particles.

3. EXPERIMENTS

3.1. Dataset and error measure

The dataset we use to evaluate the score follower is adapted from ten J.S. Bach four-part chorales, each of which is about 30 seconds long. The MIDI files were downloaded from the internet[1]. The audio files are real music performances. Each piece is performed by a quartet of instruments: violin, clarinet, tenor saxophone and bassoon. Each musician's part was recorded in isolation, while the musician listened to the others through headphones. For each piece, we explore all combinations of the individual parts to generate 4 monophonic pieces, 6 duets, 4 trios and 1 quartet together with their corresponding MIDI files. In total this gives 150 music pieces of polyphony from 1 to 4. Ground-truth alignment between MIDI and audio was manually annotated. We note that the tempo of each MIDI is constant, while the audio recordings typically contain fermata after musical phrases and tempi vary within phrases.

We use *Align Rate (AR)* as proposed in [18] to measure the alignment result. For each piece, AR is defined as the percentage of correctly aligned notes in the score. A score note is said to be correctly aligned if its onset is aligned to an audio time which deviates less than 250ms from the true audio time. This measure ranges from 0 to 1. We also propose another metric called *Average Alignment Error (AAE)*, which is defined as the average absolute difference between the aligned score position and the truth score position of each frame of the audio. The unit of AAE is the musical beat and it ranges from 0 to the maximum number of beats in the score.

The two proposed score followers are denoted as *SF-Pitch* and *SF-Chroma*, corresponding to the two observation models, respectively. For both score followers, the audio performance is decomposed into 46ms long frames with 36ms overlap. A Fourier transform is performed on each frame with a 46ms long hamming window. For a baseline comparison, we use *Scorealign*, which is an open source offline audio-score alignment system[2] based on the algorithm described in [16].

3.2. Results

Figure 2 shows the two proposed score followers, SF-Pitch and SF-Chroma, outperform the offline score aligner Scorealign on our dataset. This is promising, since offline approaches have more information available to use than online approaches. The reason that Scorealign does not perform as well as the other two is that it does not explicitly models the temporal dynamics of a performance.

Both SF-Pitch and SF-Chroma performs well. Almost all pieces have more than 80% notes correctly aligned. SF-Pitch achieves a slightly higher median and fewer outliers than SF-Chroma. This indicates the multi-pitch-based observation model is superior for this task. This is intuitive, since chroma features captures the distribution of pitch classes instead of absolute pitches. The outliers of SF-Pitch and SF-Chroma show some extremely low values. These outliers correspond to musical pieces where the score followers get lost at

[1] `http://www.jsbchorales.net/index.shtml`

[2] `http://sourceforge.net/apps/trac/portmedia/wiki/scorealign`

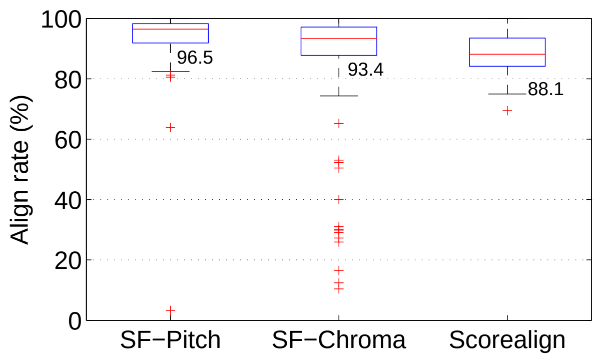

Fig. 2. Boxplots of align rate on 150 music pieces. The number besides each box indicates the median.

the beginning of the audio. Unlike offline algorithms, it is very difficult for a score follower to find the right spot, once it is lost.

Table 1. Ave±Std of align rate and average alignment error versus polyphony.

metric	poly	SF-Pitch	SF-Chroma	Scorealign
Align rate (%)	1	88.4±15.6	81.0±19.5	86.5±6.7
	2	96.3±3.3	88.5±20.3	89.1±5.7
	3	96.4±3.6	89.9±16.4	89.4±6.1
	4	96.4±3.3	92.8±6.2	88.3±7.2
AAE (beat)	1	0.40±1.22	0.79±1.47	0.22±0.07
	2	0.14±0.04	0.59±1.56	0.19±0.04
	3	0.13±0.03	0.39±1.23	0.19±0.04
	4	0.12±0.03	0.17±0.07	0.19±0.05

Table 1 presents the results for pieces with different polyphony. It is interesting to see that as polyphony increases, all three methods perform better. We believe this is because repeated notes in the monophonic pieces cause alignment errors, since note onsets are not explicitly encoded in the audio representations. Polyphonic pieces do not have this issue, since note combinations will not repeat if repeated notes appear at different score positions for different sources. Monophonic pieces with too many or too long repeated notes make the score followers lost and correspond to the outliers in Figure 2. These pieces also cause the large standard deviations for low polyphony pieces in Table 1.

The performance improvement for SF-Pitch and SF-Chroma are larger than SA-Chroma as polyphony increases, resulting the highest average align rate of 96.4% for SF-Pitch on pieces of polyphony 4. This indicates the proposed followers work well on polyphonic music. Furthermore, the average AAE of SF-Pitch for polyphony 2 to 4 are all less than a quarter beat. This would support applications like real-time score-informed source separation and pitch correction.

We provide several alignment examples of the three systems at `http://www.cs.northwestern.edu/~zdu459/icassp2011/examples`.

4. CONCLUSIONS

This paper presents an online score following approach for multi-instrument polyphonic music. Two different representations of audio are used, which results in two systems. Both systems outperform a global string alignment-based offline approach on 150 real music pieces. For future work, we would like to find another representation that can model temporal dynamics (like onsets) to improve the alignment. We would also like to apply the proposed approach to real-time score-informed source separation.

5. ACKNOWLEDGEMENT

We thank Arshia Cont for providing the code to calculate the align rate measure. We thank Craig Sapp for providing the reference of the open source software Scorealign. We also thank the three anonymous reviewers for their constructive comments. This work is supported by NSF grants IIS-0643752 and IIS-0812314.

6. REFERENCES

[1] R.B. Dannenberg, “An on-Line algorithm for real-time accompaniment,” in *Proc. ICMC*, 1984, pp. 193-198.

[2] B. Vercoe, “The synthetic performer in the context of live performance,” in *Proc. ICMC*, 1984, pp. 199-200.

[3] C. Raphael, “A Bayesian network for real-time musical accompaniment,” in *Proc. NIPS*, 2001.

[4] V. Thomas, C. Fremerey, D. Damm, and M. Clausen, “SLAVE: a score-lyrics-audio-video-explorer”, in *Proc. ISMIR*, 2009.

[5] J.J. Bloch and R.B. Dannenberg, “Real-time computer accompaniment of keyboard performances,” in *Proc. ICMC*, 1985.

[6] M. Puckette, “Score following using the sung voice,” in *Proc. ICMC*, 1995.

[7] L. Grubb and R.B. Dannenberg, “A stochastic method of tracking a vocal performer,” in *Proc. ICMC*, 1997.

[8] N. Orio and F. Déchelle, “Score following using spectral analysis and hidden markov models,” in *Proc. ICMC*, 2001.

[9] L. Grubb and R.B. Dannenberg, “Automated accompaniment of musical ensembles,” in *Proc. AAAI*, 1994, pp. 94-99.

[10] S. Ewert, M. Müller and P. Grosche, “High resolution audio synchronization using chroma onset features,” in *Proc. ICASSP*, 2009, pp. 1869-1872.

[11] S. Dixon, “Live tracking of musical performances using online time warping,” in *Proc. DAFx*, 2005, pp. 92-97.

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

[13] A. Cont, “A coupled duration-focused architecture for real-time music-to-score alignment,” *IEEE T-PAMI*, vol. 32, no. 6, pp. 974-987, June, 2010.

[14] C. Joder, S. Essid and G. Richard, “A comparative study of tonal acoustic features for a symbolic level music-to-score alignment,” *ICASSP*, 2010, pp. 409-412.

[15] Z. Duan, B. Pardo and C. Zhang, “Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions,” *IEEE T-ASLP*, in press.

[16] N. Hu, R.B. Dannenberg and G. Tzanetakis, “Polyphonic audio matching and alignment for music retrieval,” in *Proc. WASPAA*, New Paltz, New York, USA, 2003, pp. 185-188.

[17] A. Doucet, N. de Freitas and N.J. Gordon, editors, *Sequential Monte Carlo methods in practice*, Springer-Verlag, New York, 2001.

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