Event-based Multitrack Alignment using a Probabilistic Framework

A. Robertson and M. D. Plumbley

Centre for Digital Music, School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK. *Email address:* a.robertson@qmul.ac.uk

Draft of January 15, 2015

This paper presents a Bayesian probabilistic framework for real-time alignment of a recording or score with a live performance using an event-based approach. Multitrack audio files are processed using existing onset detection and harmonic analysis algorithms to create a representation of a musical performance as a sequence of time-stamped events. We propose the use of distributions for the position and relative speed which are sequentially updated in real-time according to Bayes' theorem. We develop the methodology for this approach by describing its application in the case of matching a single MIDI track and then extend this to the case of multitrack recordings. An evaluation is presented that contrasts our multitrack alignment method with state-of-the-art alignment techniques.

Introduction

The studio environment offers musicians the ability to use artificial devices such as overdubbing, editing and sequencing in order to create a recording of a musical piece. However, when they then come to perform these pieces live, such methods cannot be used. Musicians then either create an alternative arrangement that is more suited to a live rendition or they make use of backing tracks to play some of the studio parts. At present, when bands make use of this second option, the backing tracks are unresponsive to the timing variations of live performers, thereby forcing the musicians to follow the timing of the backing through use of a click track.

Automatic accompaniment is the problem of

real-time scheduling of events within a live musical performance without such constraints as using click tracks. Applications include audio synchronisation, such as the case described above where musicians require additional parts that have been overdubbed in a studio recording to play automatically during live performances, and video and lighting synchronisation, where visual aspects of the show might have been programmed relative to a rehearsed version. In both cases, an automatic accompaniment system would be expected to synchronize sufficiently accurately with the performers so that any scheduled accompaniment, either audio or visual, is perceptually 'in time'.

In the studio, it is common to record instruments separately using a dedicated microphone on each instrument channel. These individual recordings collectively constitute the multitrack, so that audio tracks for each instrument are available. There have been increasing use of multitracks both in commercial games such as Rock Band, where players attempt to 'play' each part in time with the song, and album releases allowing others to create their own remix. Techniques for the auto-

Thanks to the Royal Academy of Engineering and the EPSRC for funding this research. Thanks to Sebastian Ewert for assisting in the evaluation study and to Simon Dixon for advice on the methodology.

matic mixing of multitracks have been proposed (Reiss, 2011) which choose parameters for equalization and level with the aim of creating a professional quality stereo mix. Intelligent audio editing (Dannenberg, 2007) analyses a set of multitracks using a machine readable score to identify individual notes and help the editing process. In this paper, we examine how multitracks might be used for automatic accompaniment using a probabilistic framework. First we shall look at some of the existing methods for automatic accompaniment before examining how to go about designing a multitrack-based system for rock and pop music.

Score Following Systems

In the classical domain, this task has received considerable attention where it is often presented in the context of score following (Orio et al., 2003), the problem of aligning a performer's rendition to their location in the score. Score following systems were introduced independently at the 1984 ICMC (Dannenberg, 1984; Vercoe, 1984). These first systems used a symbolic representation of the input and made use of string matching to compare the live stream with the score. Symbolic-based matching required human supervision and commonly experienced difficulties when faced with complex events such as trills, tremolos and repeated notes (Puckette, 1992). Audio transcription and symbolic-based matching using hashing has been used to retrieve the corresponding piece and score position from a database of scores (Arzt et al., 2012).

A probabilistic method to tracking a vocal performance was introduced by Grubb and Dannenberg (1997) in which the performer's location is modeled as a probability distribution over the score. This distribution is then updated on the basis of new observations from a pitch detector. The probability that the performer is between two locations is then given by integrating the function between these two points, making explicit the uncertainty for any given alignment.

An alternative probabilistic approach is the use of graphical models, which have been employed in various forms. The hidden Markov model (HMM), successfully used in many sequential analysis tasks

such as speech recognition (Rabiner, 1989), was used by Raphael (1999) and Orio and Dechelle (2001). In both formulations, a two-level HMM is employed. One HMM level models the higher level sequence of score events such as notes, trills, rests, and the other models lower level audio features that are observed during each event, such as attack, sustain, rest. The HMM thus gives rise to a probability distribution over all the hidden states which constitute the model of the score. The *Antescofo* system (Cont, 2008) also makes use of Markovian techniques within its real-time alignment system and augments this with a tempo agent that enables the integration predictive scheduling of electronic parts within the composition process (Cont, 2011). Joder et al. (2011) propose the use of the Conditional Random Field (CRF), a graphical model structure that generalises Bayesian Networks by removing the assumption of conditional independence between observations and neighbouring hidden states. For labelling tasks, a HMM can be seen as a particular case of a CRF. A probabilistic framework using a score pointer with states identified at the level of the tatum (typically divisions of eighth or sixteenth notes) is used by Peeling et al. (2007).

One difficulty when designing such systems is incorporating a temporal model that accounts for the fact that we expect notes to last for a given duration. Raphael (2006) has investigated the use of hybrid graphical models in which both the score location and tempo are modeled as two random variables. *Antescofo* has integrated semi-Markov models into its design in which label durations are explicitly modeled. The system is reactive, allowing a high degree of flexibility to timing changes, but by modeling the current tempo, accompaniment parts can be sequenced to happen in time with anticipated events.

Otsuka et al. (2010) propose a method using a particle filter where each particle has a score position and tempo. At a fixed time step, a prediction stage updates the score positions for all particles, then an update routine ascribes a measure to to each particle according to how well it matches recent observations. This iterative process allows many hypotheses to be followed in parallel. Montecchio and Cont (2011) investigate the ability of a

particle filter to adapt to gradual and sudden tempo change. Duan and Pardo (2011) examine the use of particle filtering for score alignment using both pitch and chroma features. The methodology presented in this paper also has similarities with particle filter approaches as we employ distributions for both position and tempo and make use of prediction and update routines. An important difference is that we represent the probability distributions at a fine level of discretisation (typically 1 msec for the score position) and there is no re-sampling step required.

Cemgil et al. (2001) formulate tempo tracking in a Bayesian framework using the Kalman filter (Kalman, 1960), an efficient recursive filter used for estimating the internal state of a linear dynamic system from a series of noisy measurements. The filtering process uses two stages: prediction, in which the system's model is used to create a prediction from the last state estimate, and an update stage in which the prediction is used in combination with observation to create the new estimated state. Our proposed method also employs prediction and update steps recursively.

Audio Synchronisation

Rather than align the live audio to a representation of the score, an alternative approach to score following is to initially convert the score into audio using a MIDI synthesizer and then aligning the two audio streams (Dannenberg, 2005; Arzt et al., 2008). Dynamic Time Warping (DTW) is commonly used to find the optimal alignment between two sequences of audio features (Hu et al., 2003; Dixon, 2005; Ewert et al., 2009). The Match Toolbox (Dixon, 2005) is an online algorithm which reduces the computation time by only calculating the similarity matrix for a limited bound around the current best path.

Alignment accuracy is critical for some applications of synchronisation such as automatic accompaniment. Müller (2007) proposes an offline onset-based score-audio synchronisation method in which pitched onset events in the audio are first aligned to a score with a coarse resolution using DTW, and then a subsequent process aligns individual notes. Similarly, Niedermeyer and Widmer

(2010) improve the resolution of the DTW method using a multi-pass approach. Firstly, note onset events are identified using a coarse chroma-based alignment and those with the highest confidence are chosen to act as note anchors and the alignment path is re-estimated. Performance statistics suggest that for solo piano music, approximately 90 % of notes are aligned within 50 msec. Arzt and Widmer (2010) introduce the use of simple tempo models to improve accuracy when using synchronisation methods.

In this paper, we introduce the use of multi-tracks for the purpose of audio synchronisation. This enables reliable traditional onset detection and pitch detection on individual instrument channels to create a list of events, consisting of the event time and an associated feature such as a pitch or chroma vector. This event list is then used to perform matching to the event list derived from the recorded audio, referred to as the score. We assume that both the reference audio and the performance are available as multi-track audio stems comprising of the same number and type of tracks.

We use a probabilistic framework in order to match these higher level audio events. This is an alternative to utilizing lower-level features and matching via a graphical model formulation. There is less computation time required for higher-level event matching since the update of the distribution is less frequent. The method is well-suited to handling polyphony in cases where it is possible to derive an appropriate representation from the performance. When discretizing the temporal space for the relative position distribution, we use a high resolution, typically 1 msec intervals. Whilst we require accurate onset detection methods to do so, this has the advantage of improving the alignment accuracy.

A System for Multitrack Synchronisation in Rock and Pop music

In rock and pop music, there tends to be no score in the classical sense. However, such music often retains the same high level features such as drum patterns, chord progressions, bass lines and melodies. Gold and Dannenberg (2011) de-

Figure 1. Multitrack event-based representation for four channels: kick drum (top), bass (second), snare (third) and guitar (fourth). The pitches of the bass notes are indicated in Hertz. The guitar track shows the strength of the chromagram representation in each of the twelve bins that correspond to the chromatic notes.

scribe this area of music as falling between the extremes of the deterministic, such as classically scored music, on the one hand, and free improvised performances on the other. Such music has a semi-improvised element, but is strongly sectionalised; the tempo is approximately steady, but there are more complex rhythm patterns. They introduce the term popular music Human-Computer Music Performance Systems (HCMPS), to describe the kinds of application we are looking to design here. Whilst they envisage additional features to such a system, such as the ability to re-arrange structure on the fly, we shall be focussing solely on synchronisation between two performances where the higher level structure is identical.

For rock and pop music, although there may be variations in actual patterns and parts played, we can expect that these will happen relative to the same underlying structure as defined by bars, beats and chords. We can expect that bass and drums will constitute the rhythm section which create the foundation over which guitars and keyboards are typically played. Since drums are percussive events, for the purposes of live synchronisation, they might be sufficiently described by an event-based representation consisting of the onset time

and drum type (e.g. kick, snare, tom) rather than using precise audio features. Similarly a bass line may be sufficiently represented using the pitch and timing information of the individual notes.

The use of multiple instrument channels for matching requires that the results of different matching procedures can all be integrated within a single framework. Our system does not have an explicit score in terms of expected pitched notes and durations. Instead, we shall analyse the multitrack data to create a list of musical ‘events’ which can be considered to function as a ‘score’. We define an event as a discrete musical observation, which has a start time in milliseconds. Onset detection methods (Bello et al., 2005) offer a way to map an audio signal onto a set of time values when new musical events begin. The score is created through offline analysis of the multitrack files using onset detection and thresholding to create a list of events on each channel.

Figure 1 shows the events resulting from the analysis of four multitrack channels. For drums (kick and snare), these events simply provide the time of each event since the beginning of the recording. In the case of bass, we make use of the *yin* monophonic pitch detection algorithm

(Cheveigné & Kawahara, 2002) to provide a list of onset times and associated pitches in Hz. For guitar and other polyphonic instruments, we make use of the chromagram representation, introduced by Wakefield (1999) and based on the work of Shepherd (1964), which provides a representation of the energy found at each of the twelve notes in a chromatic scale. It has been successfully used for audio thumbnailing (Bartsch & Wakefield, 2001) and in chord detection (Pardo & Birmingham, 2002). The chromagram has also been used in DTW alignment approaches (Hu et al., 2003; Ewert et al., 2009). One useful aspect of the chromagram for these applications is that it discards timbral information, such as might be present due to the different orchestrations, but preserves information about the harmonic content that can be used to compare the two sets of audio features. For polyphonic instruments, an onset can then be characterized as a chromagram of the audio that follows the onset event. These other attributes of events, such as a pitch or a chromagram representation, are then used in the matching process to provide a measure of the extent to which one observed event matches another.

We approach the problem using a similar formulation to that employed by Grubb and Dannenberg (1997), who proposed modeling the distribution of the performer’s location in the score. To achieve a high resolution in the probabilistic framework representing score position, we opt to divide the space into discrete units at small intervals, such as 1 msec. This contrasts with most graphical model approaches, where the discretization of the space is at the level of musical objects, such as a note or chord, with a corresponding location within the score. This probability density function can be understood as quantifying our belief as to the performers location and thus peaks in the function correspond to the most likely locations in the score. Figure 2 shows how such a distribution might look in practice where the probability density function is overlaid upon a MIDI score.

Whereas Grubb and Dannenberg employ a simplifying assumption that the tempo is a single scalar value, here we make use of a separate distribution across all possible tempo values, where the

Figure 2. An example distribution displayed relative to a MIDI score.

tempo is expressed as the relative speed of the performance relative to the recorded version. Whilst their method will work well when the scalar tempo is correct, the use of a distribution quantifies the uncertainty in the estimate which is transferred to a corresponding uncertainty in the position distribution that increases in proportion to the elapsed time between observations. We are effectively able to follow multiple tempo estimates whilst also attributing a probability to each. In order to synchronize an accompaniment to a live performance, we need to continually update the two distributions, for position and tempo, after each new observation. The maximum a posteriori (MAP) estimate of the position distribution is the most likely location of the performers with the ‘scored’ (or recorded) version. When performing these computations, both the score position and the relative speed distributions are discretized. In our implementation, we have used bins of 1 msec width for score position, which allows a high resolution, and intervals of 0.01 for the relative speed distribution.

An overview of the procedure for updating the position distribution is shown in Figure 3. The system is initialized when the performance begins (at playing time zero), so we can assume there is always a prior distribution which refers to the previously observed playing time. The process can be understood by analogy to the Kalman filter, consisting of recursive estimation using two processes: prediction (time update) and update (measurement update). In the prediction step, the last state es-

timate is used to generate a prediction according to the system model and in the second step this prediction is updated using current measurement observations to generate the next state estimate. Firstly we require a prediction for the distribution at the current performance time. Secondly, we shall need to specify how to calculate the likelihood function for the observed event by matching the observed event to events in the score for the appropriate instrument. Thirdly, we then need to update the prior distribution using the likelihood function to calculate the new posterior distribution for the performer’s location.

The first of these tasks translates the distribution according to the time that has elapsed between the last update and the current event to predict the distribution. However, since there are a range of possible tempi under consideration, this will take the form of a convolution. This procedure is best understood in the context of creating the prior used to update the distributions and so we present this in the next section once the update procedure has been described. We shall now describe how to go about executing steps two and three in which the likelihood function is calculated for each new event and the posterior distribution is updated.

Update of Position Distribution

The distribution for position, $P(t)$, is a probability density function that reflects our belief as to the performer’s current location in the score, where t is the time in milliseconds from the beginning of the score. The observed onset event can be characterized as being of several types such as a simple onset, a pitched event defined by MIDI note or fundamental frequency, or by a chromagram vector. However, the principle for updating the distribution is the same and requires a distance measure describing the extent to which events are alike or ‘match’. Here, we shall use the example where the observed event is a discrete MIDI pitch. The i^{th} observed event, o_i , can be represented as a 2-tuple, (τ_i, μ_i) , where τ_i is the playing time of the event and μ_i is the MIDI pitch. We can assume that the position distribution has been updated to reflect our belief at the playing time of the currently observed event. We then wish to calculate

Figure 3. Overview of the procedure for updating the position distribution.

a likelihood function from the observed data that specifies the probability of observing this data at each time point in the score. The score consists of simple 2-tuple events with an onset time (here relative to the beginning of the score rather than the live performance) and a MIDI pitch. Let the j^{th} such recorded event, r_j , be denoted by the 2-tuple consisting of the recorded onset time, t_j , and the MIDI pitch, m_j , so that $r_j = (t_j, m_j)$. The probability of observing the given event is highest at the locations in the score where there are matching events of the same pitch.

In general, for two events of the same instrument type, we define a similarity function that takes a value between 0 and 1 that reflects the degree to which they match. Here, we specify the function to be 1 if and only if μ_i equals m_j . Let us denote the set of matching events to the event o_i as $M(o_i)$. Then this is precisely those events in the score which have identical pitch and can be defined

Figure 4. (a) The observed performed event is compared with expected event list, in this case MIDI note events. Matching events are indicated by the white boxes. (b) The likelihood function consists of a constant noise floor, with Gaussians added centred upon the matching note events. (c) The likelihood function is used to update the prior distribution (dotted) to form the new posterior distribution (solid). The resulting peak here reflects a good degree of certainty as to the performer's location.

as

$$M(o_i) = \{r_j \in R | m_j = \mu_i\} \quad (1)$$

where r_j is the recorded event with 2-tuple (t_j, m_j) and R is the set of all recorded events.

In Figure 4, we can see an example of how matching notes in the score are used to generate

a suitable likelihood function which is then used to update the posterior distribution. The likelihood function, $P(o_i|t)$, determines the probability of observing our new data given that the location in the recording is t ms. Where there are events that are strongly matching, we expect there will be peaks in

the likelihood function since these are the points in the score which we most expect to correspond to our current location, having observed the matching note data. The observed events are still subject to expressive timing, detection noise, motor noise, and therefore we model each match using a Gaussian of fixed standard deviation σ_P around the actual location in the score. For every matching event in the set $M(o_i)$, a Gaussian centred on the corresponding score location is added to the likelihood function. We also attribute a fixed quantity of noise, ν_P , to account for the possibility that the new event does not match an expected event in the recording. For example the event might be a mistake or result from a faulty detection. This gives rise to the equation:

$$P(o_i|t) = \nu_P + \frac{(1 - \nu_P)}{|M(o_i)|} \sum_{r_j \in M(o_i)} g(t, t_j, \sigma_P) \quad (2)$$

where t_j is the recorded time of event r_j measured from the beginning in milliseconds, σ_P is a constant that determines the width of the Gaussian, and the Gaussian contribution is

$$g(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right). \quad (3)$$

Then to update the prior distribution, we simply take the product with the likelihood function and normalize:

$$P(t|o_i) \propto P(o_i|t)P(t) \quad (4)$$

Once the prior is updated, we denote the time where our the position distribution is maximal as t^* , our current best estimate. Modeling the distribution over the time spanned by the whole event list would be computationally expensive. The computation of values for the distribution takes place on a region between $t^* + \rho$ and $t^* - \rho$, centred on the current best estimate, t^* . The computation of distributions is carried out only within the observation window, determined by ρ .

Prediction of the Distribution

Our update procedure for the distribution, described in the previous section, proceeded on the assumption that we had already updated the prior distribution to the current observation time. However, first a prediction step is required that updates the position distribution obtained at the last observation time, t_{n-1} , to an estimate for the position distribution at the current time, t_n , thereby providing our prior estimate for the performer's location. If the relative speed of the performance was known exactly, we could simply translate by the equivalent amount of time that has elapsed. Here, the relative speed is represented using a distribution, which reflects an inherent amount of uncertainty, so the prediction step takes into account all the possible relative speeds and the degree to which each speed is considered probable.

The elapsed time since the last observation, t_d , is $t_n - t_{n-1}$, measured in ms. Let $P_T(\tau)$ be the relative speed distribution, over the range 0 to τ_{max} . We first transform P_T into a position distribution, P_D , corresponding to this elapsed time by calculating the distribution of a delta function centred at time 0 ms according to the current speed distribution after the observed elapsed time, t_d :

$$P_D(t) = P_T\left(\frac{t}{t_d}\right). \quad (5)$$

Thus a single delta peak at relative speed 1.0 would result in a delta peak at t_d ms, as expected. We can denote the position distribution at event time t_n by P_{L_n} . In Figure 5, we show how the resulting distribution P_D appears for the a Gaussian-shaped speed distribution for different lengths of elapsed time between observations. As the time increases, the standard deviation of P_D increases proportionally. In this case, even if the position in the score at time t_{n-1} was known exactly, such as represented by a delta function, uncertainty in the tempo distribution would contribute to uncertainty in the prior when updating the position distribution at the next observation time. $P_{L_n}(t)$ then acts as the prior position distribution, $P(t)$, in the update process described in Equation 4.

To obtain the new position distribution, we convolve the resulting distribution with the position

Figure 5. Relative speed distribution (top) and the resulting convolutions with a delta function at 0 ms in the position distribution (bottom) after different elapsed intervals.

distribution at the previous observation time, $P_{L_{n-1}}$, to obtain the distribution at the new event time. So

$$P_{L_n}(t) = (P_{L_{n-1}} * P_D)(t). \quad (6)$$

Single Track Polyphonic Matching

Before proceeding to the more complex case of multitrack matching, we will examine how this method works in a test case of aligning a real-time MIDI input to a MIDI score on a single instrument channel. The intention here is to use a simpler test case to check that the method is functioning as intended before proceeding to the case of multitrack audio alignment. However, this method might also be useful in cases where MIDI input is available

such as from a keyboard or Moog piano bar.

To evaluate the algorithm’s performance we require MIDI performances whose timing differs from the score. The RWC dataset’s Classical selection (Goto et al., 2002) contains sixty one excerpts of classical music with both audio performances and the corresponding score. In order to test the algorithm on this dataset, we require a MIDI transcription of the audio recordings. A ‘warped’ version of the MIDI was made available to us by Meinard Müller, using a technique that first aligns the audio and MIDI files using chroma-onset features (Ewert et al., 2009) and then warps the MIDI file to align it with the score. Excerpts of these recordings and associated data are available

on their website ¹.

Before we can use the proposed method to carry out these tests, we need to specify the model parameters and define a process for how the relative speed distribution will be updated. In Equation 2, we set the likelihood function noise, ν_P to be 0.8 and the standard deviation of the Gaussians, σ_P , to be 100ms. Whilst at present these parameters are set by hand, in the future it might be possible to make empirical measurements to determine them. However, each parameter will be song and performer specific, so in practice this would involve using the same noise and standard deviation that has been observed on several rehearsals of a given song.

For the tempo process, ideally we would measure the time interval between corresponding notes in both performances, and calculate the ratio over a selection of such intervals and use averaging to give an estimate of the relative tempo of the two performances. However, this presupposes that we have performed accurate score following already. Thus we look to exploit the results of the note matching that is used to update the position distribution to identify the event in the score that corresponds to each observed event. For each observed note, o_i , we find the most likely matching recorded event, $\hat{r}_i$, which is the event of identical pitch for which current position probability density function is greatest. Then for each recent observed note, o_k , within a suitable timeframe (here 4 seconds), we calculate the ratio of the time interval between the two observed note events and the time interval between the two best matching notes in the score. So for each recent observed event, o_k , we create an estimate for the relative tempo, ξ_k :

$$\xi_k = \frac{\tau_i - \tau_k}{\hat{t}_i - \hat{t}_k}, \quad (7)$$

where τ_i is the time of the i^{th} observed event and $\hat{t}_i$ is the time of the associated recorded event, $\hat{r}_i$ that is the best match to o_i .

We make use of a similar Bayesian technique to update the relative tempo distribution. First we create a likelihood function as a sum of constant

offset and a Gaussian around the tempo estimate:

$$P(\xi_k|x) = \nu_T + g(\xi_k, x, \sigma_T). \quad (8)$$

Then the relative speed distribution, $P(x)$, is updated by taking the product of the prior with the likelihood function and normalising:

$$P(x|\xi_k) \propto P(\xi_k|x)P(x) \quad (9)$$

This process is carried out iteratively for all new estimates, ξ_k . This method allows the tempo estimate to respond to the strong variations in tempo that characterize classical music. One potential weakness is that if the position distribution becomes inaccurate then this will also affect the tempo process. However, the only clear alternative would be a form of tempo pulse estimation akin to beat tracking, which can prove unreliable for classical music.

When running the algorithm on the 61 files in the RWC database, we found that 47% of the notes are matched within 40 msec and 65% are matched within 100 msec. When testing an audio synchronization algorithm on audio versions of the same MIDI files, we found it to be very sensitive to variations in timbre. Thus it is difficult to provide fair comparative statistics that meaningfully compare our method with alternative systems. Software and demonstration videos of the MIDI-based matching are available for download ².

The method was observed to work best when the tempo estimate was approximately correct. When the tempo of the performed MIDI varied significantly from the tempo of the MIDI score, there was a potential for the system to become lost, particularly if there was a low density of notes. In contrast, when the pieces consisted of a high density of notes of varying pitch, the resulting distribution peaks around the correct location as there is more information to utilize. This concurs with what we might expect in a human listener, where expectation will be more accurate when the musical events are close together in time.

¹ <http://www.mpi-inf.mpg.de/resources/MIR/SyncRWC60/>

² <http://code.soundsoftware.ac.uk/projects/bayesian-multitrack-matching>

Multitrack Evaluation

To evaluate the algorithm for multitrack input, we used a collection of studio recordings of rock and pop genre songs, for which two alternative takes exist for each song. These takes are from the same recording sessions and were recorded one after the other. Some differences exist such as drum fills or change in bass line. The instrumentation included drums, bass and guitar in all cases. All were recorded without the use of click track, so the tempo was free to fluctuate. Four channels were used for the matching algorithm: bass, kick drum, snare and guitar. The offline processing, as previously described, gives rise to an event-based representation as was shown in Figure 1.

For each channel, we then provide a suitable similarity measure. For both the kick and snare drum channels, two events (on the same channel) are considered similar and the measure is 1. For bass events, we set the similarity to 1 if the pitches correspond to the same chromatic note, otherwise 0. For the guitar channel, we assign the similarity between two events by normalising each chromagram so that the maximum value is one and taking the cosine distance using the dot product between the two chroma vectors.

The parameters for the model were set by hand. The ratio of noise added, ν_P in Equation 2, was set to 0.1, 0.2, 0.6 and 0.5 for kick drum, snare drum, bass and guitar respectively. The standard deviation of the Gaussians, σ_P , was set to 6, 6, 30 and 50 ms respectively for the same instruments. The underlying motivation behind this choice is the idea that that drum events are accurately placed in time and can be used to locate precisely the point we are at in the song. In Figure 6 we can see how the likelihood function appears for a kick drum event. There are several possible matches and our low values of ν_P and σ_P result in several sharp peaks around the candidate events. The resulting posterior peaks around the most likely event. In contrast, guitar and bass events are matched using a wider Gaussian and a larger noise parameter as their intended function is to ensure we are in the correct general locality when matching the more precise drum events. In the case where there was an instrumental intro section, we allowed an ini-

tialisation procedure for our algorithm, whereby the position distribution could be set on cue to a Gaussian around a chosen point, such as the start of the verse or where the drums enter.

For our tempo process, we assume that the two performances are approximately the same speed. In view of this, we initialize a Gaussian around the relative speed ratio of 1.0 with a standard deviation of 0.1. This allows a reasonable amount of variation in tempo without any requiring any matching of high level musical features such as bars and beat. In case the two performances are at marginally different speeds, we update our estimate according to the actual synchronisation speed that is sent out as a result of matching the events in the position distribution. For each new event, we look at the inter-onset interval observations occurring on the same instrument channel. Assuming these correspond to an integer multiple of the beat interval, we calculate possible corresponding tempo observations and where each of these is close to the current estimate, we add a Gaussian around the observation. When the ratio to the current estimate is outside the range 0.9 to 1.1, we assume it to be erroneous. We then use the method described in equations 8 and 9, where the likelihood is created by adding Gaussians around these tempo observations and updating, with the parameters standard deviation σ_T set by hand to 4 msec and constant noise ν_T to 0.02.

Our implementation of the algorithm runs in a program created using openFrameworks. We use a MaxMSP patch to perform onset detection on both live input and pre-recorded files used to simulate a live performance environment. Although the program runs at approximately 20Hz, the onset events are time-stamped in MaxMSP and the alignment takes place using this accurate timing information. For each frame of the program, we update the projected alignment time and store this data. For ground truth annotations, we made use of an offline beat tracker based on the methods of Davies and Plumley (2007) in the application Sonic Visualiser (Cannam et al., 2006). These were corrected by hand to ensure the beats began at the correct point. There is an inherent ambiguity in specifying ground truth annotations. If an offline algorithmic technique is used, as in this case, the

Figure 6. The likelihood function (dotted, top) consisting of a combination of noise and narrow Gaussians centred on several matching kick events (solid lines, top) in the matching window. The posterior distribution (solid, bottom) after updating the prior (dotted, bottom) with the likelihood function.

algorithm can be subject to performance errors, so there is a limit on how accurate these can be. If humans tap in real-time to annotate various points in the audio, these can be subject to similar errors since they reflect the predicted time rather than the observed time. One other option is to annotate by hand and verify the timing, in which case specific events must be chosen that we can identify in each recording, such as the kick drum on the first beat of the bar and so on. This would constitute a non-causal descriptive annotation since these annotations describe where the beat actually occurred rather than where a human or algorithm predicted

it to be. Here we have opted for automatic annotations that were then verified manually.

Table 1 shows the results for all songs using both the offline and online techniques. Our method improves upon the Match algorithm and achieves similar errors to the Ewert et al.’s (2009) algorithm. The offline methods are provided with the endpoints of the two files as well as the start points and thus have considerable advantage over the online methods.

A mixdown of these tracks was used to allow comparison with offline methods. We created alignments of each pair of mixdowns using Match

Song Title	Median absolute alignment error (ms)			
	Online		Offline	
	Bayesian Matcher	Match OF	Ewert et al.	Match OB
Diamond White	10.7	25.0	9.2	12.5
Marble Arch	12.5	46.6	17.7	22.2
Lewes	16.4	54.1	13.8	20.7
Wanderlust	34.6	62.1	26.2	41.3
Motorcade	13.9	38.7	13.4	19.7
Festival	18.9	45.8	12.4	12.4
Station Gate	23.0	70.2	14.8	27.2
Penny Arcade	13.1	594.8	16.6	270.3
Son Of Man	13.0	28.6	13.6	14.2
New Years Resolution	15.9	75.6	13.6	24.5
Stones	12.3	40.5	12.5	19.6

Table 1

The median absolute alignment error in ms for each song.

Dixon (2005) in both the online (OF) and offline (OB) modes, and using the algorithm of Ewert et al. (2009). Since any discrepancy will be audible, we require the synchronisation to be as accurate as possible. Seeking a bound for this, Lago and Kon (2004) argue that synchronisation within the region of 20 to 30ms, equivalent to a distance of approximately ten meters, should be sufficiently accurate so as not to be perceptible. The results for all algorithms are shown in Table 1. With our proposed method, we observed that 64% of the events were recorded within 20ms of the annotated times and 89% within 40ms. These figures compare well with those achieved by Ewert et al.’s (2009) algorithm for offline audio synchronisation, the current state-of-the-art, which scored 64% and 87% for the same time limits. Our method is reliant on the presence of a significant number of percussive events. Without these, the chromagram events on their own are not sufficient to synchronise two sources and alternative methods should be employed.

Live Testing

In order to verify the results from offline tests and to experience how this interactive system might be used in practice, we also conducted tests

with a three piece rock band (bass, drums and guitar) using a total of four songs. The *elastic~* object for MaxMSP ³ which implements the z-plane timestretching algorithm ⁴ was used to modify the playing speed of the backing audio to match the system’s optimal alignment position. We also made use of marker points so that the buttons of a MIDI footpedal could set the position distribution to a Gaussian around set positions in the song, such as first verse or chorus. This proved to be a relatively unproblematic way to initialize the system successfully after a count-in or introduction section.

In all four cases the system succeeded in synchronizing backing parts in a musically acceptable way. The combination of drum and harmonic instruments allows the system to recover from situations where automatic synchronisation might be difficult, such as when there is not a steady stream of events of different type. One of the difficulties encountered when testing the system in performance is the requirement to have some kind of visual feedback of how it is behaving. Our implementation in OpenFrameworks allows the user to observe the probability density function and verify

³ Purchased from <http://www.elasticmax.co.uk/>

⁴ www.zplane.de

that the system is functioning as expected.

Conclusion

In this paper, we have presented a Bayesian probabilistic framework for the real-time alignment of a performance with a multitrack recording. Probability distributions for the position and speed of the live performance relative to the multitrack recording are updated in real-time through the sequential use of Bayes' theorem. We have observed comparable performance statistics to the use of state-of-the-art offline algorithms and confirmed that the system functions well within a live band scenario. These other algorithms were provided with stereo mixes whereas our proposed method required the multitrack audio.

The probabilistic framework allows for the integration of data from multiple sources. Providing the information can be expressed as a likelihood function for each source, it is then possible to update a global probability density function for the whole performance. The specification of a tempo distribution as well as a position distribution brings about a real-time dynamic system, in which uncertainty in the position distribution increases with the time between observations. The framework allows for the outputs of other algorithmic techniques to be used. For example, one potential development would be to incorporate beat tracking into the model. Where there is a strong beat, both tempo and position distributions might benefit from making use of the resulting tempo and phase estimates. This could be weighted according to the 'confidence' of the beat tracker. Another improvement that could be made is to model how the distribution might respond to the presence of expected events in the score which have not been observed.

Future work includes the incorporation of high-level musical knowledge. At present, the system does not have a model for rhythm, beats or bars. Reliable real-time beat tracking algorithms could improve the tempo process by comparing the observed real-time beat period to the offline beat period in the recording. Tempo induction algorithms could easily be integrated into the tempo process. Structural analysis of music might bring advan-

tages in the alignment process and such as a system would be able to provide a foundation on which generative musical systems could be created. Another potential area for development is the inclusion of training in rehearsal such as employed by Raphael (2010) and Vercoe (1985). Statistics from rehearsals could provide information such as how probable a given event is to be detected and the standard deviation in the timing. Such information could then be used when determining the likelihood function of an event in the matching procedure.

A repository containing the source code is publicly available on the Sound Software website ⁵. This includes the C++ code for the openFrameworks project and MaxMSP patches which were used to conduct the evaluations and to do live performance testing. We envisage that this can enable others to reproduce the results contained in this paper and to build upon the methods described.

References

- Arzt, A., Böck, S., & Widmer, G. (2012). Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the international conference on music information retrieval (ismir)*.
- Arzt, A., & Widmer, G. (2010). Simple tempo models for real-time music tracking. In *Proceedings of the 7th Sound and Music Computing Conference*.
- Arzt, A., Widmer, G., & Dixon, S. (2008). Automatic page turning for musicians via real-time machine listening. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*.
- Bartsch, M. A., & Wakefield, G. A. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics, 2001*.
- Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio In Processing*, 13(5, Part 2), 1035-1047.

⁵ <http://code.soundsoftware.ac.uk/projects/bayesian-multitrack-matching>

- Cannam, C., Landone, C., Sandler, M. B., & Bello, J. (2006). The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR-06)*.
- Cemgil, A. T., Kappen, H. J., Desain, P., & Honing, H. (2001). On tempo tracking: Tempogram Representation and Kalman filtering. *Journal of New Music Research*, 28(4), 259-273.
- Cheveigné, A. de, & Kawahara, H. (2002). Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4), 1917-1930.
- Cont, A. (2008). Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *Proceedings of the 2008 International Computer Music Conference*.
- Cont, A. (2011). On the creative use of score following and its impact on research. In *Proceedings of the 8th Sound and Music Computing Conference (SMC)*, Padova.
- Dannenberg, R. B. (1984). An on-line algorithm for real-time accompaniment. In *Proceedings of the 1984 International Computer Music Conference* (p. 193-198).
- Dannenberg, R. B. (2005). Toward automated holistic beat tracking, music analysis and understanding. In *Proceedings of the International Conference on Music Information Retrieval* (p. 366-373).
- Dannenberg, R. B. (2007). An intelligent multi-track audio editor. In *Proceedings of the international computer music conference* (p. 89-94).
- Davies, M. E. P., & Plumbley, M. D. (2007). Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3), 1009-1020.
- Dixon, S. (2005). Match: A music alignment tool chest. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05)* (p. 492-497).
- Duan, Z., & Pardo, B. (2011). A state space model for online polyphonic audio-score alignment. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing* (p. 197-200).
- Ewert, S., Müller, M., & Grosche, P. (2009). High resolution audio synchronization using chroma onset features. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*.
- Gold, N., & Dannenberg, R. B. (2011). A reference architecture and score representation for popular music human-computer music performance systems. In *Proceedings of the 2011 International Conference on New Interfaces for Musical Expression* (p. 36-39).
- Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2002). Rwc music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)* (p. 287-288).
- Grubb, L., & Dannenberg, R. B. (1997). A stochastic method of tracking a performer. In *Proceedings of the 1997 International Computer Music Conference* (p. 301-308).
- Hu, N., Dannenberg, R. B., & Tzanzetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the 2003 International Computer Music Conference* (p. 185-188).
- Joder, C., Essid, S., & Richard, G. (2011). A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Transactions on Audio, Speech and Language Processing*, 19(8), 2385-2397.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. In *Transaction of the amse-journal of basic engineering* (p. 35-45).
- Lago, N. P., & Kon, F. (2004). The quest for low latency. In *Proceedings of the 2004 International Computer Music Conference* (pp. 33-36).
- Montecchio, N., & Cont, A. (2011). A unified approach to real time audio-to-score and audio-to-audio alignment using sequential montecarlo techniques. In *Proceedings of the 2011 International Conference on Audio Speech and Signal Processing (ICASSP 2011)* (p. 193-196).
- Müller, M. (2007). *Information retrieval for music and motion*. Springer.
- Niedermeyer, B., & Widmer, G. (2010). A multi-pass algorithm for accurate audio-to-score alignment. In

- Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR), Utrecht* (p. 417-422).
- Orio, N., & Dechelle, F. (2001). Score following using spectral analysis and hidden Markov models. In *Proceedings of the 2001 International Computer Music Conference* (p. 105-109).
- Orio, N., Lemouton, S., & Schwarz, D. (2003). Score Following: State of the art and new developments. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression* (p. 36-41).
- Otsuka, T., Nakadai, K., Takahashi, T., Komatani, K., Ogata, T., & Okuno, H. G. (2010). Design and implementation of two-level synchronization for an interactive music robot. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence, AAAI 2010* (p. 1238-1244).
- Pardo, B., & Birmingham, W. (2002). Improved Score Following for Acoustic Performers. In *Proceedings of the 2002 International Computer Music Conference*.
- Peeling, P., Cemgil, A. T., & Godsill, S. (2007). A probabilistic framework for matching music representations. In *Proceedings of the 8th international conference on music information retrieval (ismir 2007)* (p. 267-272).
- Puckette, M. (1992). Score following in practice. In *Proceedings of 1992 International Computer Music Conference* (p. 182-185).
- Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.
- Raphael, C. (1999). A probabilistic expert system for automatic accompaniment. *Journal of Computational and Graphical Statistics*, 10(3), 467-512.
- Raphael, C. (2006). Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning*, 21(4), 360-370.
- Raphael, C. (2010). Music Plus One and machine learning. In *Proceedings of the 27th International Conference on Machine Learning*.
- Reiss, J. D. (2011). Intelligent systems for mixing multitrack audio. In *Proceedings of the International Conference on Digital Signal Processing, DSP2011* (p. 1-6).
- Shepherd, R. (1964). Circularity in judgements of relative pitch. *Journal of the Acoustical Society of America*, 36, 2346-2353.
- Vercoe, B. (1984). The Synthetic Performer in the context of live performance. In *Proceedings of the 1984 International Computer Music Conference* (p. 199-200).
- Vercoe, B., & Puckette, M. (1985). Synthetic Rehearsal, training the Synthetic Performer. In *Proceedings of the 1985 International Computer Music Conference* (p. 275-278).
- Wakefield, G. H. (1999). Mathematical representation of joint time-chroma distributions. In *Proceedings of the SPIE Conference on Advanced Signal Processing Algorithms, Architectures and Implementations* (pp. 637-645).