Aligning Musical Scores with Audio using Hybrid Graphical Models

Abstract

A method is presented for aligning a polyphonic musical score to an audio performance of that score. The method is based on a graphical model containing both discrete hidden variables, corresponding to score position, and continuous hidden variables, corresponding to tempo. The data interpretation is defined to be the most likely configuration of the hidden variables and the computation of this configuration is developed. Applications to digital music libraries and musical accompaniment systems are presented.

Introduction

We address an audio recognition problem in which we develop a correspondence between a musical score and an audio performance of that score. There are two versions of this problem which we call "on-line" and "off-line" parsing. Off-line parsing uses the complete performance to estimate the onset time for each score note. Applications of off-line parsing include editing and post-processing of digital audio, as well as allowing "random access" to music — playback starting at arbitrary *musical* location. Our personal interest in this problem is motivated by work in musical accompaniment systems which spawns another application of off-line parsing. This application as well as work done for the Variations2 Digital Libary Project are discussed in the "Applications" section.

On-line parsing processes the data in real-time, so no "look ahead" is possible. The goal here is to identify the musical events depicted in the score with as little latency as possible. Musical accompaniment systems must perform this task with the live soloist's input. Other applications include the automatic coordination of audio-visual equipment with musical performance, such as opera supertitles and real-time score-based audio enhancement e.g. pitch correction. We will treat the off-line problem in this work, however extensions to on-line parsing are possible.

Many researches have treated on-line and off-line musical score alignment including (Dannenberg 1984), (Vercoe 1984), (Puckette 1995), (Grubb & Dannenberg 1997), (Raphael 1999), (Orio & Dechelle 2001), (Turetsky & Ellis 2003), (Soulez, Rodet, & Schwarz 2003). While many variations exist, the predominant approach develops an objective function measuring the quality of a particular match between score and data. Then the best possible match is sought using dynamic programming. Without doubt, these efforts contain many notable successes, however, the problem is still open. Our personal experience indicates that one needn't look far (e.g. complex polyphony, varied instrumental texture, fast notes, rearticulations and octave slurs, etc.), to find audio data that confounds any particular algorithm. The need for a more robust and widely applicable algorithm is the motivation for the current work.

An "Achilles' heel," common to all past approaches we know, is the modeling of length for the individual notes. If the issue is treated at all, note lengths are either constrained to some range or modeled as random, with the range or distribution depending on a global tempo or learned from past examples. Either implicitly or explicitly, the note lengths are treated as independent variables. However, note lengths are anything but independent. Our belief, bolstered by conventional musical wisdom, is that the lion's share of note length variation can be explained in terms of a time-varying tempo process. This work explicitly models tempo as a real-valued process, hoping that the more powerful model will be able explain what the data model cannot.

While the remaining discussion is directed entirely toward our particular application, we believe that the basic technique we introduce is generally applicable to time series analyses in which periodic state-conditional observations are made on a process whose state-change times follow linear dynamics.

The Model

In the case of monophonic (single voice) music, a musical score can be represented as a sequence of score positions, expressed in beats, with associated pitches. Polyphonic music can be viewed, similarly, as a sequence of score positions with associated *chords* (pitch collections). In the polyphonic case, the score positions would be created by sorting the union of score positions over all musical parts, and discarding duplicate positions. Each score position would then be associated with the collection of pitches that sound until the next musical event. Thus our score does not represent which notes come from which instruments. We notate this score by $(m_1, c_1), (m_2, c_2), \ldots, (m_K, c_K)$ where the kth event be-

gins at m_k beats and contains pitches $c_k = (c_k^1, \ldots, c_k^{L_k})$. By convention, $m_1 = 0$ and c_K is a 0-note "chord" corresponding to the silence at the end of the audio.

Let $t_1, \ldots, t_k$ be the sequence of times, in seconds, at which the chord onsets occur. If the music were performed strictly in time, then $t_k = sm_k + t_1$ and the score matching problem would reduce to estimating the two parameters, s and t_1. However, such robotic precision is seldom desired and never achieved in actual human performance. More typically the onset times are the product of numerous interpretative issues as well as the inevitable inaccuracies of human performance. We model here what we believe to be the most important factors governing musical timing: time-varying tempo as well as note-by-note deviations. More precisely, we define random variables $T_1, \ldots, T_K$ and $S_1, \ldots, S_K$ through

$$\begin{aligned} S_k &= S_{k-1} + \sigma_k \\ T_k &= T_{k-1} + l_k S_k + \tau_k \end{aligned}$$

for $k = 2, \ldots, K$ where $l_k = m_k - m_{k-1}$ is the length, in beats, of the kth chord, and $\{S_1, T_1\} \cup \{\sigma_k, \tau_k\}_{k=2}^K$ are independent normal random variables with the $\{\sigma_k, \tau_k\}_{k=2}^K$ variables further assumed to be 0-mean. $S_1, \ldots, S_K$ is our tempo process, modeled as a random walk, where S_k is the local beat length (secs. per beat) at the kth chord. The $T_1, \ldots, T_K$ are the actual note onset times which depend on the tempo process as well as the deviations $\{\tau_k\}_{k=2}^K$. We view the actual times $t_1, \ldots, t_K$ as a realization of the random variables $T_1, \ldots, T_K$. The dependency structure of these variables is expressed as a directed acyclic graph in the top of Figure 1.

Letting $s = (s_1, \ldots, s_K)$ and $t = (t_1, \ldots, t_K)$, this model leads to a simple factorization of the joint probability density, $p(s, t)$, as

$$p(s,t) = p(s_1)p(t_1) \prod_{k=2}^{K} p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k)$$

The factors in this equation are, more explicitly,

$$\begin{aligned} p(s_1) &= N(s_1; \mu_{s_1}, \sigma_{s_1}^2) \\ p(t_1) &= N(t_1; \mu_{t_1}, \sigma_{t_1}^2) \\ p(s_k|s_{k-1}) &= N(s_k; s_{k-1}, \sigma_{s_k}^2) \\ p(t_k|t_{k-1}, s_k) &= N(t_k; t_{k-1} + l_k s_k, \sigma_{t_k}^2) \end{aligned}$$

$k = 2 \ldots, K$, where $N(\cdot, \mu, \sigma^2)$ denotes the univariate normal density function with mean μ and variance σ^2. The model parameters $\mu_{s_1}, \sigma_{s_1}^2, \mu_{t_1}, \sigma_{t_1}^2$ and $\{\sigma_{s_k}^2, \sigma_{t_k}^2\}_{k=2}^K$ are assumed known.

Our data come from a sampled audio signal which we partition into a sequence of frames, $y_0, y_1, \ldots y_{N-1}$, each corresponding to Δ seconds of sound ($\Delta \approx 30$ ms. in our experiments). For each frame, $n = 0, \ldots, N-1$, we let x_n denote the *index* of the chord that is sounding for the frame. For example, the sequence of values:

$$x_0, x_1, x_2 \ldots = \underbrace{00\ldots0}_{I_0}\underbrace{11\ldots1}_{I_1}\underbrace{22\ldots2}_{I_2}\ldots \qquad (1)$$

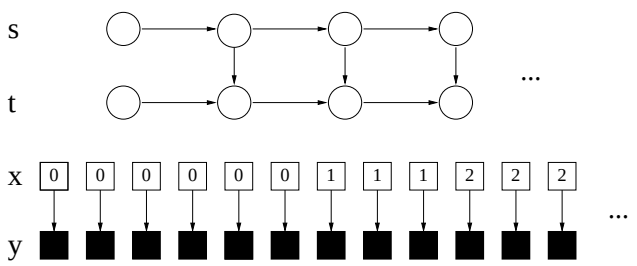

Figure 1: The dependency structure of the model variables expressed as a directed acyclic graph (DAG). Circles denote continuous variables, while squares denote discrete variables. Darkened circles represent observed variables.

would signify that the first note begins at $I_0\Delta$ seconds, the second note begins at $(I_0 + I_1)\Delta$ seconds, etc. We model both $x = (x_0, \ldots, x_{N-1})$ and $y = (y_0, \ldots, y_{N-1})$ as realizations of random processes X and Y. In particular, given $X = x$, $Y_0, \ldots, Y_{N-1}$ are conditionally independent random vectors with local dependence on the X process. That is, the conditional density for Y given $X = x$ is

$$p(y|x) = \prod_{n=0}^{N-1} p(y_n|x_n)$$

The X and Y processes are depicted in the bottom of Figure 1.

We are interested in specifying a *joint* model on the vectors S, T, X, Y. The key observation here is that, up to a discretization error, T and X contain *identical* information: the partitioning of the audio signal into chords. Thus we can eliminate t from the model and write $p(s, t, x, y) = p(s, x, y)$ where

$$\begin{aligned} p(s,x,y) &= p(s_1)p(t_1) \prod_{k=2}^{K} p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k) \\ &\times \prod_{n=0}^{N-1} p(y_n|x_n) \end{aligned} \qquad (2)$$

While the right hand side of Eqn. 2 appears to depend on t, we interpret $t = t(x)$, i.e.

$$t_k(x) = \min\{n : x_n = k\}\Delta \qquad (3)$$

The Data Model

Our data model gives $p(y_n|x_n)$ where y_n is the nth frame of audio data and x_n is an index into the score. Suppose first that x_n indexes a single-note chord, whose frequency is c. Since musical pitches are nearly periodic, we would expect the power spectrum of y_n to concentrate energy primarily at the integral multiples $c, 2c, 3c, \ldots$. Thus we build a probability density on frequency by

$$g(\omega; c) = \sum_{j=1}^{J} p_j N(\omega; jc, \tau_j^2)$$

where $\sum_{j=1}^{J} p_j = 1$. The number of harmonics considered, J, their widths, $\{\tau_j^2\}_{j=1}^{J}$, and the way in which they decay, $p_1, \ldots, p_J$, are parameters of our model. This can be extended to an arbitrary chord, $c = (c^1, \ldots, c^L)$, by mixing the one-note distributions (and adding a background noise term)

$$g(\omega; c) = q_0 1_{(0,\omega_{\max})} + \frac{q_1}{L} \sum_{l=1}^{L} g(\omega, c^l)$$

where q_0, q_1 are weights with $q_0 + q_1 = 1$.

Suppose our observed power spectrum is $e = e(y_n) = (e_1, \ldots, e_F)$. Imagining for a moment that the values of e are integral, we can view e as the histogram of a random sample of size $r = \sum_f e_f$ from our model distribution $g = (g_1, \ldots, g_F)$ where g is a discretization of $g(\omega; c)$. Thus

$$p(e|c, r) = \frac{r!}{\prod_f e_f!} \prod_{f=1}^{F} g_f^{e_f} \tag{4}$$

To complete the data model we would need a marginal distribution on r, $p(r)$. However, since the data, e, will be fixed in our computations, $p(r)$, or any other factor not depending on g amounts to a multiplicative constant as c varies in Eqn. 4. Thus or model is

$$p(y_n|x_n) = p(e|c) = C(e, \alpha) \prod_{f=1}^{F} g_f^{e_f/\alpha}$$

where we have dropped the requirement that e be integral and added a scaling parameter α.

The parameters of our model, $J, \{p_j, \tau_j^2\}_{j=1}^{J}, \alpha, q_0, q_1$ should ideally be learned from training data, although they were set by hand in the present case.

Computing the MAP Estimate

Our goal is now phrased as follows: Given the observed data, y, we seek the most likely configuration of the unobserved variables, s and x:

$$(\hat{s}, \hat{x}) = \arg\max_{s,x} p(s, x, y) \tag{5}$$

If all variables were discrete, this problem would be solvable through traditional dynamic programming techniques, however, note that the s process is continuous. We describe here a method for approximating the global solution to Eqn. 5.

Consider the tree of Figure 2, which describes an enumeration of all possible realizations of the labeling process x. First, the root of the tree is labeled 0. Then, any node in the tree with label $k < K$ will have two children labeled by k and $k + 1$, while a node labeled K will have a single child labeled K. The labels 0 and K correspond to the silence at the beginning and end of the audio data. Note that any path from the root of the tree to a node at depth n traverses a sequences of labels corresponding to a possible realization of the $x_0, \ldots, x_n$; clearly all possible realizations are contained in the tree.

Traversing a partial path through the tree (fixing $x_0, \ldots, x_n$) implicitly determines the first several variables

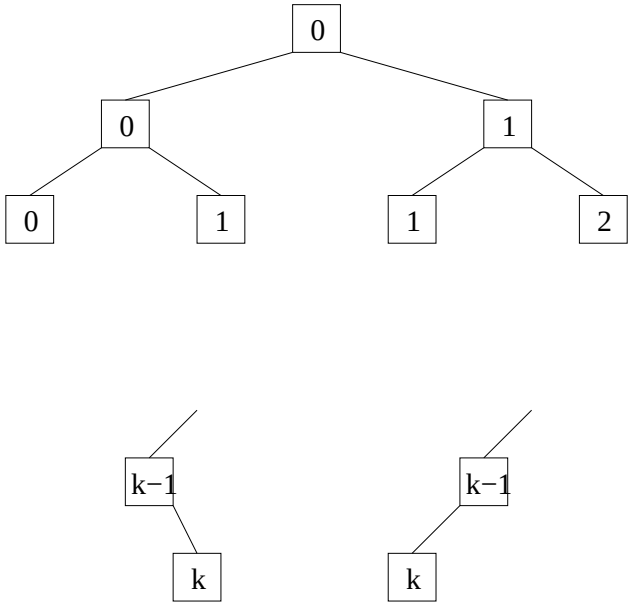

Figure 2: The search tree associated with optimization. The label of a tree node is the score index describing the current chord.

$t_1, \ldots, t_k$ through Eqn. 3. (Here k also depends on the path $x_0 \ldots, x_n$ although we suppress this in our notation.) For this partial path, the probability density for the variables $X_0^n = (X_0, \ldots, X_n)$, $Y_0^n = (Y_0, \ldots, Y_n)$, $S_1^k = (S_1, \ldots, S_k)$ is

$$\begin{aligned} p(s_1^k, x_0^n, y_0^n) &= p(s_1)p(t_1) && (6) \\ &\times \prod_{\kappa=2}^{k} p(s_\kappa|s_{\kappa-1})p(t_\kappa|t_{\kappa-1}, s_\kappa) && (7) \\ &\times \prod_{\nu=0}^{n} p(y_\nu|x_\nu) \end{aligned}$$

In interpreting this equation, if $k = 0$ then the lines numbered 6 and 7 are unity while if $k = 1$ only the line numbered 7 is unity.

For each partial path x_0^n define

$$\hat{p}_{x_0^n}(s_k) = \max_{s_1 \ldots, s_{k-1}} p(s_1^k, x_0^n, y_0^n) \tag{8}$$

In right side of Eqn. 8, all variables are fixed except $s_1, \ldots, s_k$. Due to the Gaussian assumptions, $p(s_1^k, x_0^n, y_0^n)$ is clearly the exponential of a quadratic function of the $s_1, \ldots, s_k$. Thus, after maximizing over $s_k \ldots, s_{k-1}$, we can represent $\hat{p}_{x_0^n}(s_k)$ as a function of the parametric form

$$K(s; h, m, v) = he^{-\frac{1}{2}(s-m)^2/v} \tag{9}$$

for some (h, m, v).

It is a simple matter to compute the parameters of this representation recursively. Suppose $x_0^{n-1} = (x_0, \ldots, x_{n-1})$ is such that $x_{n-1} = k - 1$ and $\hat{p}_{x_0^{n-1}}(s_{k-1}) =$

$K(s_{k-1}; h, m, v)$. There are two cases. First, if $x_{n-1} = x_n$ then

$$\begin{aligned}\hat{p}_{x_0^n}(s_{k-1}) &= \hat{p}_{x_0^{n-1}}(s_{k-1})p(y_n|x_n) \\ &= K(s_{k-1}; hp(y_n|x_n), m, v)\end{aligned}$$

Otherwise, a straightforward but somewhat lengthy calculation gives

$$\begin{aligned}\hat{p}_{x_0^n}(s_k) &= \max_{s_{k-1}} \hat{p}_{x_0^{n-1}}(s_{k-1})p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k) \\ &\times\ p(y_n|x_n) \qquad (10) \\ &= K(s_k; h', m', v')\end{aligned}$$

where

$$\begin{aligned}h' &= \frac{hp(y_n|x_n)}{2\pi\sigma_{s_k}^2\sigma_{t_k}^2} e^{-\frac{1}{2}\frac{(t_k - t_{k-1} - l_k m)^2}{l_k^2(v+\sigma_{s_k}^2)+\sigma_{t_k}^2}} \\ m' &= \frac{m\sigma_{t_k}^2 + l_k(t_k - t_{k-1})(v + \sigma_{s_k}^2)}{l_k^2(v + \sigma_{s_k}^2) + \sigma_{t_k}^2} \\ v' &= \frac{(v + \sigma_{s_k}^2)\sigma_{t_k}^2}{l_k^2(v + \sigma_{s_k}^2) + \sigma_{t_k}^2}\end{aligned}$$

In the sequel we will use the notation $\hat{p}_{x_0^n}(s_k) = K(s_k; h(x_0^n), m(x_0^n), v(x_0^n))$. Thus for any complete path $x = x_0^{N-1}$ with $x_{N-1} = K$, we can compute

$$\max_s p(s, x, y) = \max_{s_K} \hat{p}_{x_0^{N-1}}(s_K) = h(x_0^{N-1})$$

We now discuss maximizing $p(s, x, y)$ over x as well as s.

Controlling the Search Tree's Growth

Clearly the number of tree nodes is exponential in the tree's depth making it impossible to explore the tree thoroughly without additional insight. The key observation here is the the tree can be pruned without sacrificing the search for the optimal path.

Examination of Eqn. 2 reveals that our joint probability model has a Markov structure and can hence be factored into "past" and "future" probabilities. That is, if $x_n = k$ we have

$$\begin{aligned}p(s, x, y) &= p(s_1^k, x_0^n, y_0^n) \\ &\times\ p(s_{k+1}^K, x_{n+1}^{N-1}, y_{n+1}^{N-1}|s_k, t_k(x_0^n))\end{aligned}$$

where k, s_k, t_k plays the role of the "state." Maximizing both sides over s yields

$$\begin{aligned}\max_s p(s, x, y) &= \max_{s_k^K} \hat{p}_{x_0^n}(s_k) \\ &\times\ p(s_{k+1}^K, x_{n+1}^{N-1}, y_{n+1}^{N-1}|s_k, t_k(x_0^n))\end{aligned}$$

Now consider two paths in the tree, x_0^n and $\tilde{x}_0^n$, both beginning the kth note on the nth frame, as depicted in Figure 2, so that $t_k(x_0^n) = t_k(\tilde{x}_0^n) = n\Delta$. Suppose that $\hat{p}_{x_0^n}(s_k) > \hat{p}_{\tilde{x}_0^n}(s_k)$ for all s_k. Then for any possible future evolution of the path: x_{n+1}^{N-1},

$$\max_s p(s, x_0^n, x_{n+1}^{N-1}, y) > \max_s p(s, \tilde{x}_0^n, x_{n+1}^{N-1}, y)$$

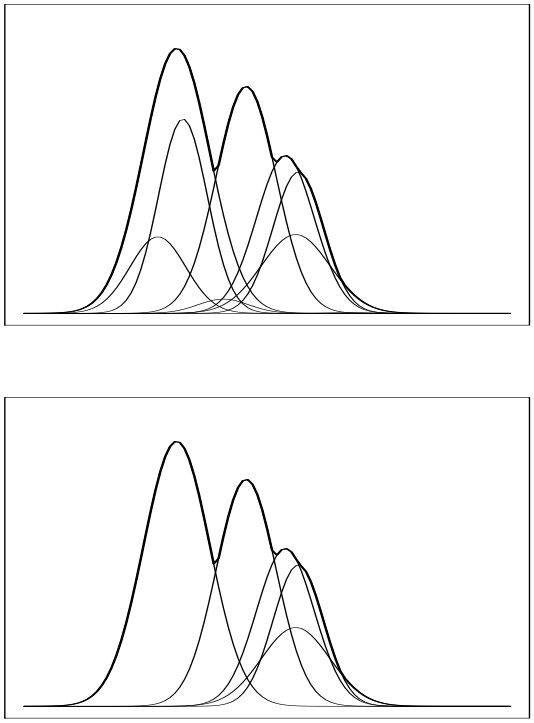

Figure 3: **Top:** The set of functions $\{\hat{p}_{x_0^n}(s_k) = K(s_k; h(x_0^n), m(x_0^n), v(x_0^n)) : x_0^n \in I\}$ (i.e. before thinning). **Bot:** The functions $\{\hat{p}_{x_0^n}(s_k) : x_0^n \in \mathrm{Thin}(I)\}$

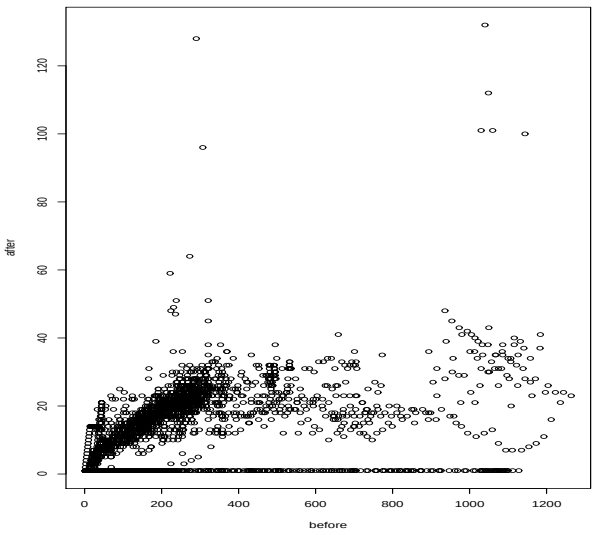

Figure 4: Each point in figure represents the number of hypotheses, both before and after the thinning operation. The resulting number of hypotheses (y) seems independent of the number thinned (x).

Thus any path beginning with $\tilde{x}_0^n$ can always be beaten by a path beginning with x_0^n and we can prune the former branch no danger of losing the optimal path.

More generally, suppose that I is some collection of paths such that for $x_0^n = (x_0, \ldots, x_n) \in I$, $x_{n-1} = k - 1$ and $x_n = k$. Define the set $\mathrm{Thin}(I)$ as the smallest subset of I such that

$$\max_{x_0^n \in \mathrm{Thin}(I)} \hat{p}_{x_0^n}(s_k) = \max_{x_0^n \in I} \hat{p}_{x_0^n}(s_k)$$

as in Figure 3. Reasoning as above, we can prune any partial path $x_0^n \notin I$ without loss of optimality. Due to the simple parametric form of the $\hat{p}_{x_0^n}(s_k)$ functions, the thinning procedure can be computed with a computational cost that is quadratic in $|I|$. An algorithm for the thinning operation is discussed in (Raphael 2002).

Suppose I_{N-1} is the set of branches, $x = x_0^{N-1}$, surviving through the final iteration $N - 1$. Then

$$\begin{aligned} \max_{s,x} p(s, x, y) &= \max_{x_0^{N-1} \in I_{N-1}} \max_{s_K} \hat{p}_{x_0^{N-1}}(s_K) \\ &= \max_{x_0^{N-1} \in I_{N-1}} h(x_0^{N-1}) \end{aligned}$$

Thus the maximizing path is $\hat{x} = \arg\max_{x_0^{N-1} \in I_{N-1}} h(x_0^{N-1})$ and the maximizing sequence of tempos can be found by letting $\hat{s}_K = m(\hat{x})$ and tracing the optimal tempos $\hat{s}_{K-1} \ldots, \hat{s}_1$ backward by recursively substituting $\arg\max$ for $\max$ in Eqn. 10.

Computational Complexity and Pruning

We examine here the computational complexity of computing a global optimum.

Suppose that

1. The thinning algorithm reduces any set of branches to most T branches.
2. The number of chords in the score is K.
3. Each note can last no longer than F frames.

Then at the beginning each iteration, n, the number of surviving branches is at most KTF since each chord could have begun at frames $n - 1, \ldots n - F$ with each (chord, onset position) pair associated with at most T branches. The basic iteration extends the surviving branches and thins each subset of children that begin chord k, $k = 1, \ldots, K$. Thus the cost of an iteration is $KTF + K(TF)^2$ since the thinning operation is quadratic in the number of branches to be thinned. Thus the overall computational complexity of the algorithm is $NK(TF)^2$.

As a coarse approximation the first assumption seems justifiable in practice. Figure 4 shows a scatter plot where each point is the number of hypotheses both before and after thinning. The "after"numbers seem to be independent of the "before" numbers and very rarely does thinning result in more than 50 branches. In fact, the number would be less at coarser frame discretization. However, as the algorithm progresses, the number of (chord, onset position) hypotheses entertained in any frame increases monotonically since thinning happens only within a (chord, onset position) collection. Eventually this creates an overwhelming number of hypotheses, so global optimization is only possible for short scores containing 30 or so notes.

While global optimization may not be possible, we strongly doubt that it is necessary. Human listeners certainly lose sight of the distant past while following musical scores, yet have no trouble maintaining an accurate correspondence between score and sound. We proceed analogously by pruning our search tree. Recall that the basic iteration computes $\hat{p}_{x_0^n}(s_k) = K(s_k; h(x_0^n), m(x_0^n), v(x_0^n))$. One possibility is to prune the branches after sorting on $h(x_0^n) = \max_{s_k} p_{x_0^n}(s_k)$. The problem with this method is that the number of chords encountered, k varies from branch to branch so the resulting scores are products of different numbers of factors. In particular, in experimenting with this pruning method, we have observed that branches already exceeding a reasonable amount of time for the current chord avoid being pruned by delaying the chord change and the associated note length factor $p(t_{k+1}|t_k, s_k)$. This phenomenon is analogous to the "horizon effect" of computer chess in which hypotheses receive falsely inflated scores by postponing an inevitable end beyond the search horizon. We address this issue by sorting instead on

$$\tilde{h}(x_0^n) = \max_{s_k, t_{k+1} > n\Delta} p_{x_0^n}(s_k) p(t_{k+1}|t_k, s_k)$$

We still are comparing apples and oranges to some extent, but the actual results are much improved. The results of the next section were produced keeping a total of several hundred hypotheses for each frame.

Applications

Part of our interest in this problem stems from a collaboration with the Variations2 Digital Music Library Project at

Indiana University. One of the many aims of this project is to allow listeners, in particular students in their School of Music, new tools for learning and studying music, interleaving sound, text, music notation, and graphics. One specific goal is to give the user "random access" to a recording allowing playback to begin at any time, expressed in musical units, e.g. the third beat of measure 47. Clearly such a task requires an "index" into the audio recording labeling the times of the many musical events, such as that produced by our algorithm.

We construct the score corresponding to an audio file automatically by analyzing a MIDI file of the same composition. MIDI is a symbolic representation of music specifying, among other things, the start time, end time, and pitch for each note in the musical score. From many MIDI files, such as those generated from score-writing programs, it is possible to reconstruct the musical times (in beats or measures) of these events in a straight-forward manner. Further processing leads to our score representation as a sequence of labeled chords $(m_1, c_1), (m_2, c_2) \ldots$. The MIDI file also provides approximate tempos for the various sections of the composition, as well as meter changes.

Our score matching technique was first applied separately to the *Adagio Sostenuto* and *Vivace* from the 1st movement of Beethoven's 7th Symphony. A formal evaluation of the results requires tedious and error-prone hand-marking of a large amount of music, and is beyond the scope of our current research goals. Rather, to informally evaluate the results, we have superimposed clicks identifying important musical positions on the audio files. These were placed on every bar line and half-measure in the Adagio and on every bar line in the Vivace. The audio files can be heard at http://fafner.math.umass.edu/aaai04. With the exception of two momentary problems caused by the two *fermati* in the Vivace, we believe the correspondence is quite accurate, and certainly good enough for the needs of the application.

The second, and much more challenging, example is the Sacrificial Dance from Stravinsky's *Le Sacre du Printemps*. This composition challenges the score-following abilities of many educated musicians with its frequent meter changes and complicated relationship between barring and musical content. We prepared our score in a fully automatic way from a MIDI file of a piano reduction involving many simplifications of the original score. The composition contains several sudden tempo changes which were handled by resetting our tempo to the value indicated in the MIDI file at the appropriate location (and letting it vary according to the model elsewhere), thus allowing us to parse the entire Sacrificial Dance as a single section. This example can also be heard at the referenced web page with a more complicated regimen for adding clicks: the 3/16, 1/8, and 3/8 bars get 1 loud click on the downbeat; 5/16 bars are treated as 3+2 with one loud click on the downbeat and one weaker click on the 4th 16th; 1/4 bars are treated similarly as 2+2; 2/4, 3/4, and 5/4 bars get 1 one loud click on the downbeat and weaker clicks on the other quarter note pulses. The web page also contains a piano score for comparison. Careful study and familiarity with the piece are necessary to evaluate the results, however, we believe they are also rarely off by more than a fraction of a beat. Our own experience informally evaluating these results leads us to conclude that the music index we develop is quite useful for learning a difficult score.

The current work was motivated by a different application, namely our efforts in building a musical accompaniment system. Our system generates a full orchestral accompaniment that follows a live player in real time, as in a concerto. While an overview of this work is beyond the present scope, one fundamental task is to create an index into an audio recording of the accompaniment, to be used in the real-time resynthesis of the audio. The web location referenced above contains a click file showing the alignment achieved by our algorithm on a section of the Brahms Violin Concerto (without the soloist). For this application it is still necessary to occasionally adjust occasional onset times given by our algorithm; the Brahms example shows the results before any hand-tuning takes place. A recording of a live performance of part of the work, with the orchestra played by our accompaniment system, can be heard at http://fafner.math.umass.edu/music_plus_one.

Acknowledgments

Thanks to Don Byrd and members of the Variations2 Digital Music Library Project at Indiana University for discussions regarding practical aspects of this work.

References

Dannenberg, R. 1984. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference, 1984*, 193–198. Int. Computer Music Assoc.

Grubb, L., and Dannenberg, R. 1997. A stochastic method of tracking a vocal performer. In *Proc. Int. Comp. Music Conf.*, 301–308. Int. Computer Music Assoc.

Orio, N., and Dechelle, F. 2001. Score following using spectral analysis and hidden markov models. In *Proc. Int. Comp. Music Conf.*, 151–154. Int. Computer Music Assoc.

Puckette, M. 1995. Score following using the sung voice. In *Proc. Int. Comp. Music Conf.*, 175–178. ICMA.

Raphael, C. 1999. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Trans. on PAMI* 21(4):360–370.

Raphael, C. 2002. A hybrid graphical model for rhythmic parsing. *Artificial Intelligence* 137(1):217–238.

Soulez, F.; Rodet, X.; and Schwarz, D. 2003. Improving polyphonic and poly-instrumental music to score alignment. In *Proc. Int. Symp. Music Info. Retrieval*, 143–150. ISMIR.

Turetsky, R., and Ellis, D. 2003. Ground-truth transcriptions of real music from force-aligned midi syntheses. In *Proc. Int. Symp. Music Info. Retrieval*, 135–142. ISMIR.

Vercoe, B. 1984. The synthetic performer in the context of live performance. In *Proceedings of the International Computer Music Conference, 1984*, 199–200. Int. Computer Music Assoc.