Aligning Music Audio with Symbolic Scores Using a Hybrid Graphical Model*

Christopher Raphael
School of Informatics
Indiana University, Bloomington
craphael@indiana.edu

September 19, 2005

Abstract

We present a new method for establishing an alignment between a polyphonic musical score and a corresponding sampled audio performance. The method uses a graphical model containing both latent discrete variables, corresponding to score position, as well as a latent continuous tempo process. We use a simple data model based only on the pitch content of the audio signal. The data interpretation is defined to be the most likely configuration of the hidden variables, given the data, and we develop computational methodology to identify or approximate this configuration using a variant of dynamic programming involving parametrically represented continuous variables. Experiments are presented on a 55-minute hand-marked orchestral test set.

1 Introduction

We address an audio recognition problem in which a correspondence is established between a polyphonic musical score and an audio performance of that score, known as score alignment or score matching. There are two versions of this problem, often called "on-line" and "off-line" recognition.

Off-line parsing uses the complete performance to estimate the onset time for each score note, thus the off-line problem allows one to "look into the future" while establishing the match. Part of our interest in off-line parsing 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 the 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 this application requires either hand marking of audio or off-line parsing. Another off-line application is the editing and post-processing of digital audio, in which many tasks require the location or processing of a specific place in a potentially large audio datafile. Additional interest in score

*This work supported by NSF grants IIS-0113496 and IIS-0534694.

alignment comes from the desire to study musical expression using audio data; clearly, score alignment is needed before the relevant information can be extracted from the data. Our personal interest in off-line parsing is motivated by yet another application: our work in musical accompaniment systems. In this effort we resynthesize a prerecorded audio performance at variable rate to accompany a live soloist. The synchronization requires that we begin with a correspondence between the prerecorded audio and a musical score. In fact, almost any application requiring the syncronization of a data stream or media with prerecorded audio requires score alignment.

On-line parsing, sometimes called *score-following*, processes the data in real-time as the signal is acquired. Thus, no "look ahead" is possible and the data must be processed in real-time. The goal of on-line parsing is to identify the musical events depicted in the score with little latency and high accuracy. 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.

Many researchers have treated on-line and off-line musical parsing including [1], [2], [3], [4], [6], [7], [8], [5], [9], [10], [11], to name several. While many variations exist, the predominant approach seeks a best possible match by "warping" the score to fit the data using some form of dynamic programming. The measures of match quality are quite varied, including edit-like distances and probabilistic measures, as in the popular hidden Markov model approaches. Without doubt, these efforts contain many notable successes, however, the problem still remains open. In our personal experience with the HMM approach cited above, results degrade, sometimes dramatically, as we encounter increasingly difficult domains such as complex polyphony, varied instrumental texture, fast notes, rearticulations (repeated notes with same pitch) and octave leaps, large tempo changes, unpitched sounds, etc. While the literature contains very little in the way of formal evaluations, other researchers seem to experience similar problems. The need for a more robust and widely applicable approach is the motivation for the current work.

We believe the "Achilles' heel" of all past approaches we know, including our own, is the modeling of length for the individual notes. If the issue is treated at all, note lengths, or, more precisely, inter-onset intervals, 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 regarded 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. The failure to model time-varying tempo shifts more burden to the audio data modeling, requiring the method to follow the score almost exclusively using sound, without regard for one of the most basic aspects of musical timing. For instance, a model that knows only the global tempo, as opposed to *time-varying* tempo, must allow the recognized note lengths to deviate far from what the tempo and printed note values would predict, since tempo changes will produce significant departures from the note lengths implied by the global tempo. On the other hand, a model that simultaneously tracks tempo will make reasonable predictions of note onset times, even when the audio data is ambiguous, as in repeated chords. This work explicitly models tempo as a real-valued process, hoping that the more powerful model will guide the recognition through regions of ambiguity, as well as imposing some rhythmic regularity on the recognition results. Our data model, introduced in Section 2, is indeed simple-minded, focusing exclusively on pitch content. While we expect that improvements to our system will be achieved by strengthening this model, the results presented in Section 4 argue that our focus on the tempo model is well-placed.

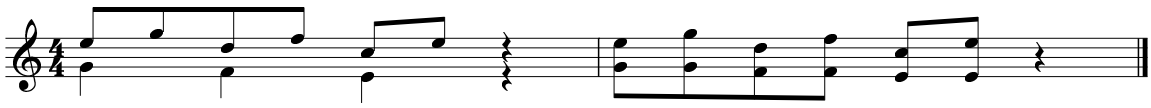

Figure 1: **Left:** The original score. **Right** The score viewed as a sequence of chords.

The most straightforward approach to tempo modeling would represent the "state" of the performance as a score position and tempo pair — both discrete variables for the sake of simplicity. From frame to frame the position would be updated using the current tempo while the tempo would be allowed to gradually vary. We have attempted such an approach using a HMM framework, but found that the discretization of position and tempo needed to be extremely fine before useful results were achieved. This earlier effort is, by no means, a "straw man" created only to motivate the current approach. Rather, our current approach stems from a deeper understanding of the computational issues learned from this previous effort.

While addressing a different musical problem, this work has a natural connection with several AI approaches to music recognition [12],[13],[14], [15]. The first two of these efforts examine the problem of beat-tracking — following the pulse of piece of music over time. Both use generative graphical models that treat tempo as a continuous latent Gaussian process, as we do. The second two efforts treat additionally the problem of musical transcription also using a graphical model with continuous latent tempo variables. A significant point of departure between our work and these is in the choice of optimization techniques. These approaches use a combination of Markov Chain Monte Carlo (MCMC) and sequential Monte Carlo techniques [17]. In contrast, we have used a dynamic programming technique which, in small examples, computes the globally most likely data interpretation. In more realistic cases, we approximate this global optimum with pruning.

We first present in Section 2 a mathematical model that combines a note-level model for rhythmic interpretation with a frame-by-frame data model. The note-level model explicitly represents tempo variation and note-by-note deviations. The data model is based completely on the pitch content of the audio. Section 3 introduces a dynamic-programming-like method that identifies the globally most likely configuration of the discrete note onset times and continuous tempo variables. The method is only feasible for small scores and pruning techniques for larger scores are discussed. Section 4 presents results on a 55-minute widely varied test set of short orchestral movements containing examples from Mozart to Shostakovich. These are, to our knowledge, the first systematic experiments of score matching in a realistic ensemble context. The results demonstrate the accuracy of our note onset estimates and show that the algorithm rarely becomes irrecoverably lost. Our dataset has been made publicly available to facilitate comparisons. This section also compares with results derived from an HMM-based method for score alignment that does not model time-varying tempo.

While the remaining discussion is directed entirely toward our particular application, we believe that the basic technique we introduce is more generally applicable. Perhaps the idea of most general interest is our framework which combines a "frame-based" model of audio, or other one-dimensional data stream, with an event-based model whose state-change times follow linear dynamics.

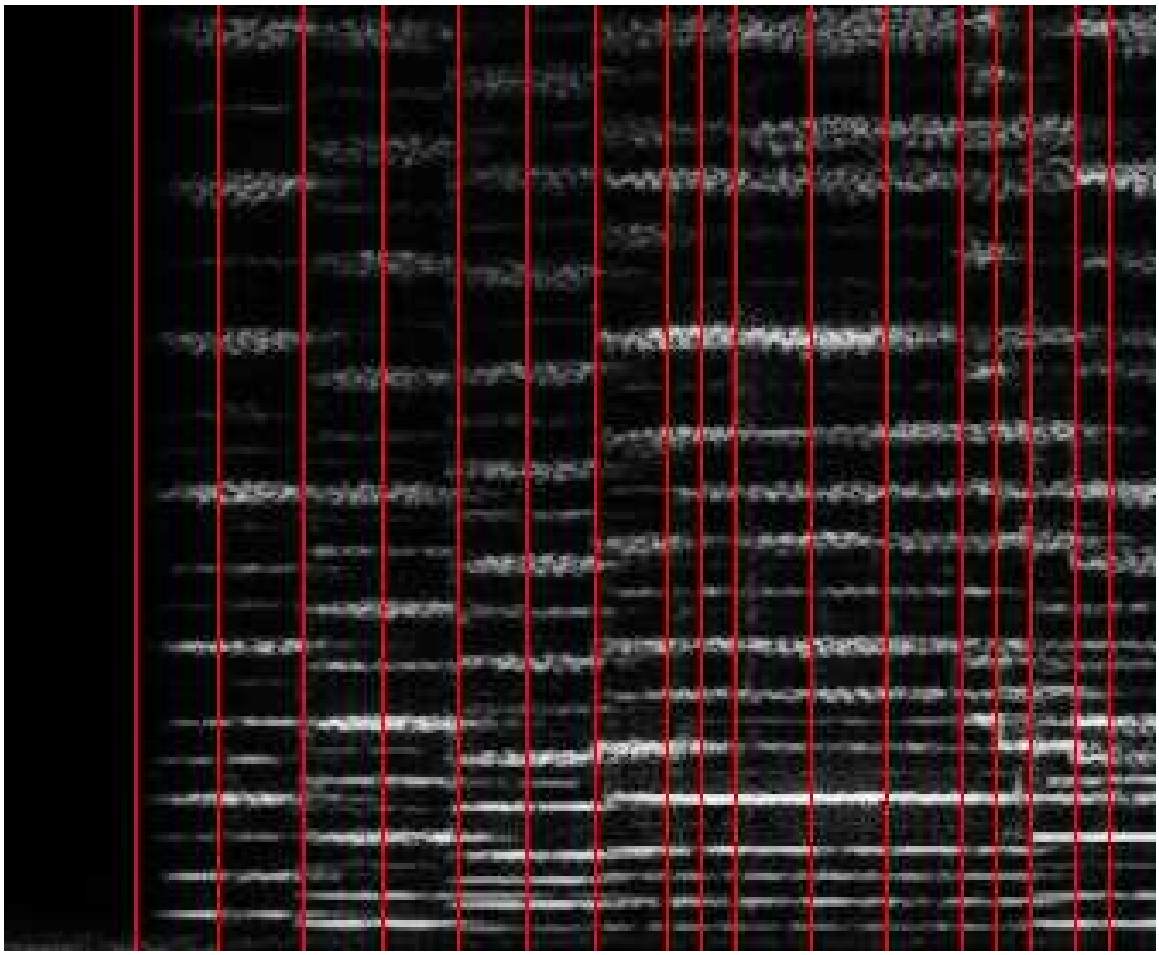

Figure 2: A spectrogram of the opening of the 2nd movement of the Mozart Oboe Concerto. The chord onset times $\{t_k\}$ are depicted as vertical lines in the figure.

2 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. We do not assume that score positions are equally spaced. 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 as depicted in Figure 1. In this figure, the left measure is the original score, while the right measure shows our "homophonic" view of this score. Thus our score does not represent which notes come from which instruments, or which notes are newly entering ("attacking"). We notate this score by

$$(m_1, c_1), (m_2, c_2), \ldots, (m_K, c_K) \tag{1}$$

where the kth event begins 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. For illustration, figure 2 depicts the $\{t_k\}$ as horizontal lines in a "spectrogram." 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 where s is the global tempo and t_1 is the time of the first musical event. 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 from what the local tempo and printed note length would predict. 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 3.

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

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

The factors in this equation are, more explicitly,

$$\begin{aligned} p(s_1) &= N(s_1; \mu_{s_1}, \sigma_{s_1}^2) && (3) \\ p(t_1) &= N(t_1; \mu_{t_1}, \sigma_{t_1}^2) && (4) \\ p(s_k|s_{k-1}) &= N(s_k; s_{k-1}, \sigma_{s_k}^2) && (5) \\ p(t_k|t_{k-1}, s_k) &= N(t_k; t_{k-1} + l_k s_k, \sigma_{t_k}^2) && (6) \end{aligned}$$

$k = 2 \ldots, K$, 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). See Figure 4. 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:

$$x_0, x_1, x_2 \ldots = \underbrace{00\ldots0}_{I_0}\underbrace{11\ldots1}_{I_1}\underbrace{22\ldots2}_{I_2}\ldots \tag{7}$$

would signify that the first chord begins at $I_0\Delta$ seconds, the second chord 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

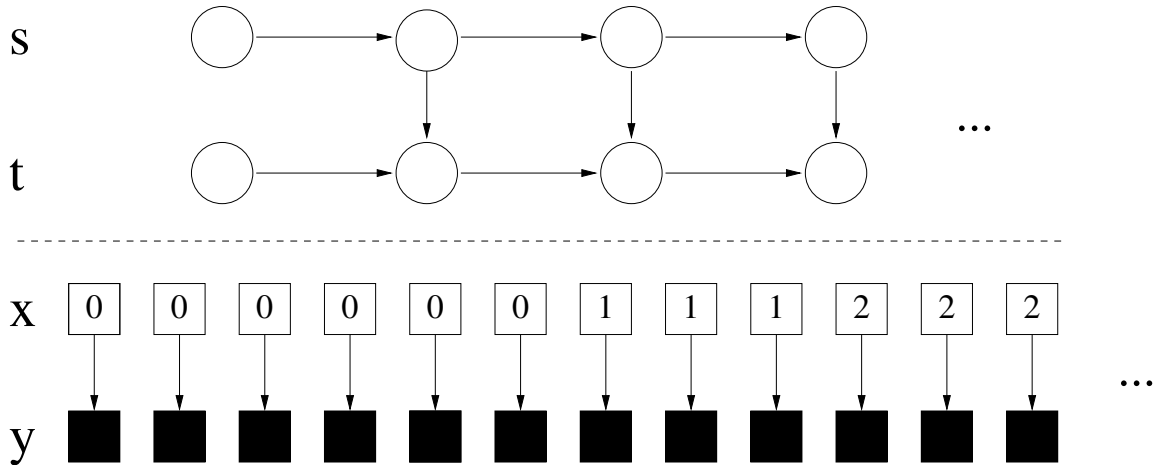

Figure 3: Top: The dependency structure of the tempo and note onset time process. Bottom: The model connecting the spectrogram data to the frame labels. Circles denote continuous variables, while squares denote discrete variables. The darkened squares represent observed variables — the spectrogram frames.

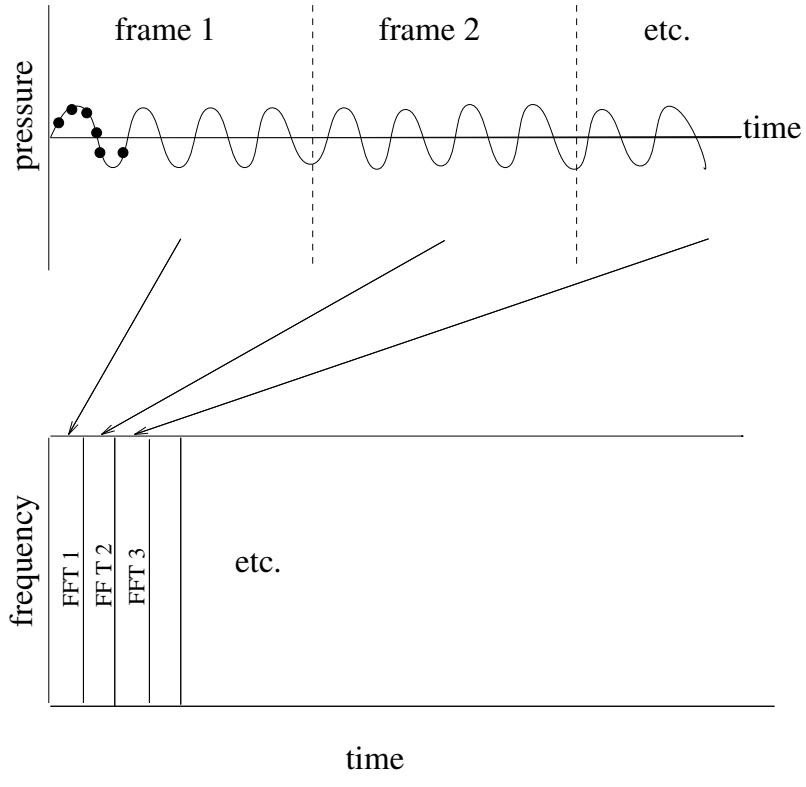

Figure 4: The data is segmented into a sequence of frames. The spectrogram is formed by taking a windowed finite Fourier transform of each frame

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 3.

We are interested in specifying a *joint* model on the vectors S, T, X, Y. The key observation here is that T and X contain essentially identical information: the partitioning of the audio signal into chords. More explicitly, if the definitions of $p(t_1)$ and $p(t_k|t_{k-1}, s_k)$ are modified to be Gaussian, *conditioned on the assumptions that* $T_{k-1} < T_k$ and that $T_k = n\Delta$ for $n \in \{1, 2, 3 \ldots\}$, then T and X are equivalent in the sense that either can represented as a function of the other. Thus, regarding t as increasing and discrete, we can eliminate t from the model and write $p(s, t, x, y) = p(s, x, y)$ where

$$p(s, x, y) = p(s_1)p(t_1) \prod_{k=2}^{K} p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k) \prod_{n=0}^{N-1} p(y_n|x_n) \tag{8}$$

While the right hand side of Eqn. 8 appears to depend on t, we interpret $t = t(x)$, i.e.

$$t_k(x) = \min\{n : x_n = k\}\Delta \tag{9}$$

As before, the $\{t_k\}$ are simply the chord onset times, in seconds, which can be directly read off the label process, x.

2.1 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 the audio signals of 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

$$g(f; c) = \sum_{j=1}^{J} w_j N(f; jc, \rho_j^2)$$

where f indexes frequency and $\sum_{j=1}^{J} w_j = 1$. The number of harmonics considered, J, their widths, $\{\rho_j^2\}_{j=1}^{J}$, and the way in which they decrease as frequency increaes, $w_1, \ldots, w_J$, are parameters of our model. The left panel of Figure 5 shows such a density. This idea 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)

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

where q_0, q_1 are weights with $q_0 + q_1 = 1$, as shown in the right panel of Figure 5.

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 $g(f; c)$. Thus

$$p(e|c, r) = \frac{r!}{\prod_f e_f!} \prod_{f=1}^{F} g_f^{e_f} \tag{10}$$

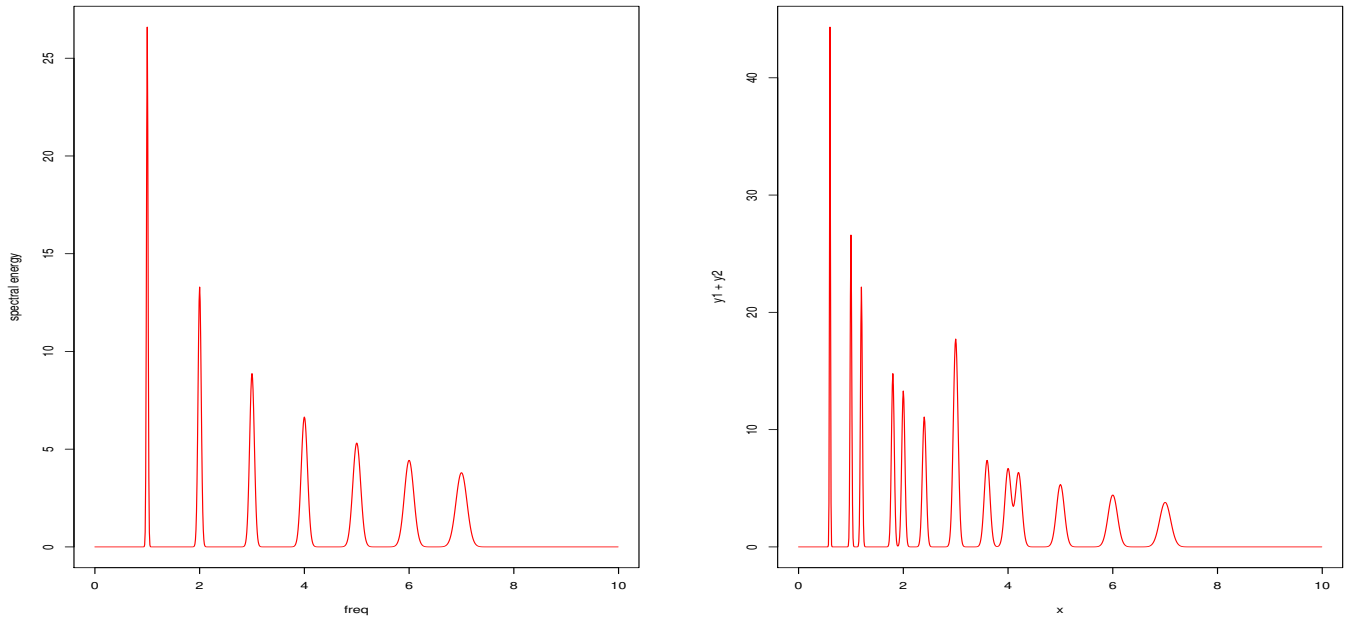

Figure 5: **Left:** An idealized spectrum for a single note. **Right:** An analogous spectrum for two simultaneous notes created as a superposition of two single-note spectra.

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 in the data model as the chord hypothesis, c, varies. 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.

While suppressed in the notation, our data model is parametric with parameters $J, \{w_j, \rho_j^2\}_{j=1}^J, \alpha, q_0, q_1$. The data model could easily be improved by adding features measuring other attributes such as local energy and "burstiness" of the signal, however, the simpler model described above was used in our experiment.

3 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:

$$(\hat{s}, \hat{x}) = \arg\max_{s,x} p(s, x, y) \tag{11}$$

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, and in some cases computing, the global solution to Eqn. 11.

Symbol	Meaning
m_k	position in beats of kth chord
c_k	the kth chord
c_k^i	the ith member of kth chord
t_k	time in seconds of kth chord onset
s_k	local tempo in seconds per beat at the kth chord
l_k	length of kth chord in beats
Δ	the length of a frame of audio in secs.
x_n	label of nth frame of audio ($x_n = k$ means nth frame in kth note)
y_n	the nth frame of data (a vector)
g_f	the energy for freq. f in a modeled spectrum
e_f	the observed energy at freq. f.

Table 1: Table of Symbols

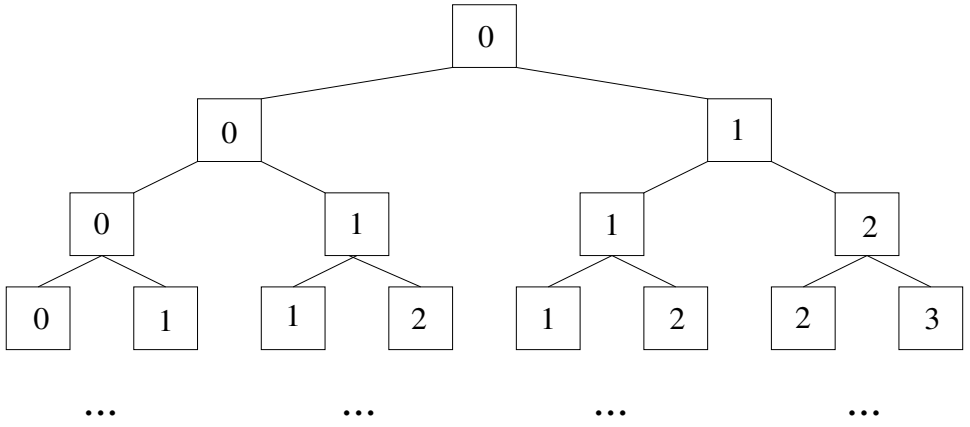

Figure 6: The search tree associated with optimization. The label of a tree node is the score index describing the current chord.

Consider the tree of Figure 6, 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 sequence 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 onset variables $t_1, \ldots, t_k$ through Eqn. 9. (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

$$p(s_1^k, x_0^n, y_0^n) \quad = \quad p(s_1)p(t_1) \tag{12}$$

$$\times \quad \prod_{\kappa=2}^{k} p(s_\kappa|s_{\kappa-1})p(t_\kappa|t_{\kappa-1},s_\kappa) \tag{13}$$
$$\times \quad \prod_{\nu=0}^{n} p(y_\nu|x_\nu)$$

In interpreting this equation, if $k=0$ then the lines numbered 12 and 13 are unity while if $k=1$ only the line numbered 13 is unity.

For each partial path x_0^n define

$$\hat{p}_{x_0^n}(s_k) = \max_{s_1\ldots,s_{k-1}} p(s_1^k, x_0^n, y_0^n) \tag{14}$$

In right side of Eqn. 14, 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

$$K(s_k;h,m,v) = he^{-\frac{1}{2}(s_k-m)^2/v} \tag{15}$$

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

$$\begin{aligned}\hat{p}_{x_0^n}(s_k) &= \max_{s_{k-1}} \hat{p}_{x_0^{n-1}}(s_{k-1})p(s_k|s_{k-1})p(t_k|t_{k-1},s_k)p(y_n|x_n)\\ &= K(s_k;h',m',v')\end{aligned} \tag{16}$$

where

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

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

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

The preceding shows how to maximize $p(s,x,y)$ over s when x is held fixed. We now discuss maximizing $p(s,x,y)$ over x as well as s.

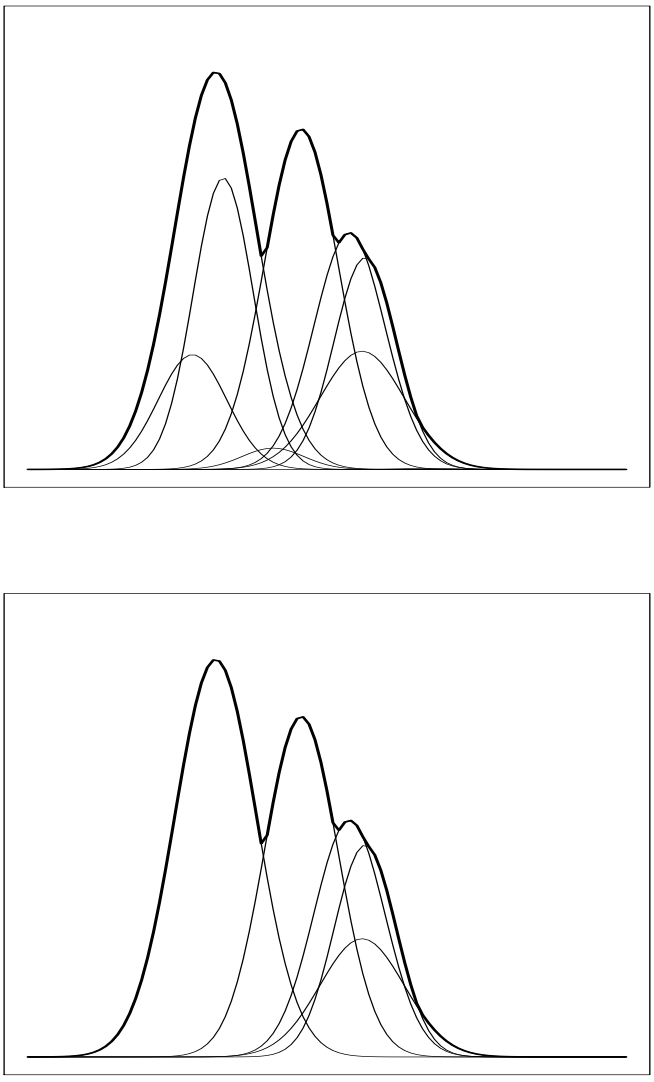

Figure 7: **Top:** The set of functions $\{\hat{p}_{x_0^n}(s_k) = K(s_k; h(x_0^n), m(x_0^n), v(x_0^n)) : x_0^n \in J_{n,k}\}$ (i.e. before thinning). **Bottom:** The functions $\{\hat{p}_{x_0^n}(s_k) : x_0^n \in \text{Thin}(I_{n,k})\}$ (after thinning). The dark maximizing line is the function $\hat{p}_{n,k}$.

3.1 Controlling the Search Tree's Growth

Clearly the number of tree nodes grows exponentially with the tree's depth, making it impossible to explore the tree thoroughly without additional insight. The key observation here is that the tree can be pruned without sacrificing the search for the optimal path, using a dynamic programming technique.

Examination of Eqn. 8 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

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

where (s_k, t_k) plays the role of the "state." In other words, given the past, the future only depends on (s_k, t_k).

The basic idea of dynamic programming observes that if $(\tilde{s}_1^k, \tilde{x}_0^n)$ is such that $\tilde{x}_n = k$ and

$$p(\tilde{s}_1^k, \tilde{x}_0^n, y_0^n) < \max_{\substack{s_1^k : s_k = \tilde{s}_k \\ x_0^n : t_k(x_0^n) = t_k(\tilde{x}_0^n)}} p(s_1^k, x_0^n, y_0^n) \tag{17}$$

then $(\tilde{s}_1^k, \tilde{x}_0^n)$ cannot be the prefix of an optimal path. Similarly, if $\tilde{x}_0^n$ satisfies Eqn. 17 *for all* $\tilde{s}_1^k$, then $\tilde{x}_0^n$ cannot be a prefix for the optimal x. In this case, $\tilde{x}_0^n$ corresponds to a partial path in the tree of Figure 6 that we could prune without any loss of optimality.

We show here how to identify such suboptimal branches. Let

$$\begin{aligned} J_{n,k} &= \{x_0^n : x_n = k, x_{n-1} = k-1\} \\ &= \{x_0^n : t_k(x_0^n) = n\Delta\} \end{aligned}$$

Define

$$\begin{aligned} \hat{p}_{n,k}(\tilde{s}_k) &= \max_{\substack{s_1^k : s_k = \tilde{s}_k \\ x_0^n \in J_{n,k}}} p(s_1^k, x_0^n, y_0^n) \\ &= \max_{x_0^n \in J_{n,k}} \hat{p}_{x_0^n}(\tilde{s}_k) \\ &= \max_{x_0^n \in I_{n,k}} \hat{p}_{x_0^n}(\tilde{s}_k) \end{aligned}$$

where $I_{n,k}$ is defined as the smallest subset of $J_{n,k}$ such that the above equation holds. A computationally feasible algorithm for the "thinning" operation which computes $I_{n,k}$ is discussed in detail in [16]. The thinning operation is depicted in Figure 7. Intuitively, any Gaussian kernel that does not "touch the skyline" is beaten by some other kernel for every possible tempo value. Thus, such a kernel can be pruned without loss. In this way we "thin" the collection of possible paths we need to consider in search of the optimal path.

If $\tilde{x}_0^n \in J_{n,k} \setminus I_{n,k}$, then for any $\tilde{s}_k$

$$\begin{aligned} p(\tilde{s}_1^k, \tilde{x}_0^n, y_0^n) &\leq \hat{p}_{\tilde{x}_0^n}(\tilde{s}_k) \\ &< \max_{x_0^n \in I_{n,k}} \hat{p}_{x_0^n}(\tilde{s}_k) \\ &= \max_{x_0^n \in J_{n,k}} \hat{p}_{x_0^n}(\tilde{s}_k) \\ &= \max_{\substack{s_1^k : s_k = \tilde{s}_k \\ x_0^n : t_k(x_0^n) = t_k(\tilde{x}_0^n)}} p(s_1^k, x_0^n, y_0^n) \end{aligned}$$

and can thus be pruned without loss.

Of course the algorithm does not need to perform the thinning operation over the entire set $J_{n,k}$. In fact, since $|J_{n,k}|$ grows exponentially and the thinning algorithm is quadratic in the size of $|J_{n,k}|$, thinning would be infeasible in this case. However, once a branch has been identified as suboptimal, no extension of the branch can become optimal at a later stage. Thus our algorithm needs only to propagate branches that have not yet been shown to be suboptimal and to, for each iteration n, perform thinning among the intersection between these surviving branches and $J_{n,k}$ for each k. It is easy to show that after the surviving subset of $J_{n,k}$ has been thinned, no further thinning is possible on children of these branches until some branches begin new chords (enter $J_{n',k+1}$ for some n'). For this reason, the thinning operations described above are the only ones that we perform.

Suppose I_{N-1} is the set of branches, $x = x_0^{N-1}$, surviving through the final iteration $N-1$ and having $x_{N-1} = K$. 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. 16.

3.2 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 at 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 of 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 8 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 chords.

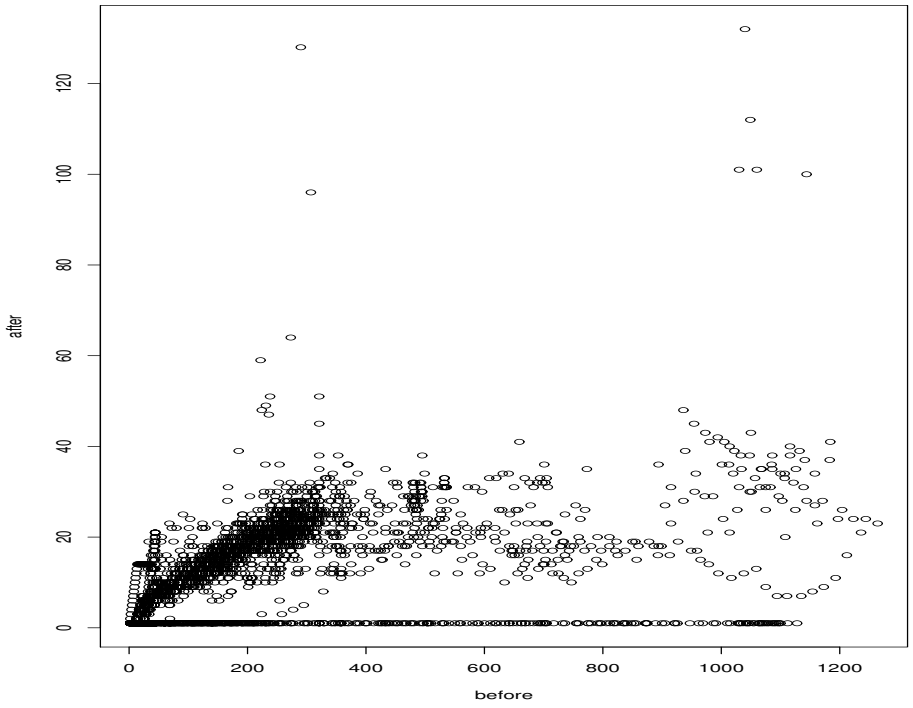

Figure 8: Each point in figure represents the number of hypotheses, both before and after the thinning operation. The resulting number of branches (y) seems independent of the number thinned (x).

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.

A simple approach would be to sort the current (surviving) hypotheses on $h(x_0^n) = \max_{s_k} \hat{p}_{x_0^n}(s_k)$ and prune the smallest of these. We have found this technique works reasonably well. However, 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, thereby delaying incurring 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. A second problem is that, at any particular analysis frame, n, the various partial paths, $x = (x_0, \ldots, x_n)$, will represent different positions in the musical score. Suppose that a particular path is currently in the kth note in the score. Then the likelihood of this partial path will contain a factor for each of the $t_1, \ldots, t_k$ as in Eqn. 2. Since k varies over the partial paths, the h-scores are composed of different numbers of factors and direct comparison is suspect.

We address these problems by sorting the partial paths over $S(x_0^n) = D(x_0^n) + \frac{Kn}{N}\frac{M(x_0^n)}{k}$ and pruning the paths having the lowest $S(x_0^n)$ scores, where

$$\begin{aligned} D(x_0^n) &= \sum_{\nu=0}^{n} \log p(y_\nu|x_\nu) \\ M(x_0^n) &= \max_{s_1^{k+1}, t_{k+1}>n} \log\{p(s_1)p(t_1)\prod_{\kappa=2}^{k+1} p(s_\kappa|s_{\kappa-1})p(t_\kappa|t_{\kappa-1}, s_\kappa)\} \end{aligned}$$

In the above equation, $D(x_0^n)$ is simply the data log likelihood of the partial path. $M(x_0^n)$ is the optimal model log likelihood for the first $k+1$ tempo and position variables with t_{k+1} taking some value in the future. Since a partial path, x_0^n, implicitly fixes the first k position variables $t_1, \ldots, t_k$, we only maximize over the remaining variables. While we omit the calculation, one can easily compute $M(x_0^n)$ recursively as n increases. At any fixed iteration, then we are sorting over the data log likelihood plus a constant times the *average per note* model log likelihood, therefore not penalizing the paths with more notes. However, as $n \to N$ and $k \to K$ this gradually reduces to the original log likelihood $D(x_0^n) + M(x_0^n)$ as in Eqn. 14.

4 Experiments

To evaluate our algorithm we constructed a test set of short orchestral movements (and one opera selection), representing a variety of musical styles. The restriction to the orchestral domain does not reflect a known limitation of our methods — to the contrary, orchestral music is quite heterogeneous and contains many of the data types we believe to be most problematic for score matching, such as tempo changes, rubato, fast notes, and varied instrumental texture. The choice of data was influenced by personal taste.

Recall that our musical score is represented as a sequence of (musical time, chord) pairs as in Eqn. 1. In many cases, this representation can be constructed automatically from a MIDI file. To this end, we collected around 20 MIDI files from the *Classical Midi Archives*. In creating our scores we replaced note sequences within a single voice that appeared to be trills or tremolos by sustained

Work	Orchestra	Conductor	Year	Mins.
Mahler				
Symphony 4 Mvmt. 1	Boston	Ozawa	1987	5.23
Holst				
The Planets Mercury	Toronto	Davis	1986	4.03
The Planets Mars	Toronto	Davis	1986	6.88
Mozart				
Symph. 41 Mvmt 2	Berlin	von Karajan	1978	7.78
Symph. 41 Mvmt 4 (1)	Berlin	von Karajan	1978	2.20
Symph. 41 Mvmt 4 (2)	Berlin	von Karajan	1978	3.84
Cosi Fan Tutte Overture	London	Haitink	1987	4.54
ibid. "Soave Sia il Vento"	London	Haitink	1987	3.02
Debussy				
Trois Nocturnes Fêtes	Cleveland	Boulez	1995	6.52
Dvorak				
Symphony 8 Allegretto	London	Leppard	1997	6.05
Shostakovich				
Symphony 1 Mvmt 2	Chicago	Bernstein	1989	4.88

Table 2: The test set for the score matching experiments.

notes. In addition, two notes very nearly sharing the same onset time are both assumed to begin at the "simpler" musical time (the one with the smaller denominator, when expressed in beats). Aside from these special cases, the processing consists of a straightforward extraction of data from the MIDI files. In particular, our algorithm creates, for each MIDI file, a note list containing the musical onset times with the corresponding MIDI note commands, a list of tempo changes, and list of meter changes. The tempo changes, discussed later, will be used for setting approximate tempi.

About half of these files were rejected for various reasons, either before or after this preprocessing stage: some files were piano reductions, some had suspiciously complex reconstructed rhythm suggesting expressive timing, some contained other assorted anomalies. There is no reason to assume that our matching algorithm would fail on the MIDI files we rejected. In fact, a previous experiment showed excellent results with a piano transcription of the Sacrificial Dance from Stravinsky's *Le Sacre du Printemps* applied to a full orchestral performance. However, our goal here was to keep the experimental conditions as constant as possible over the test set. Despite this goal, the accepted MIDI files were not of uniform quality. Some contain many wrong notes, some contain numerous tempo changes while others only mark sudden and significant tempo changes, and other sources of variability probably exist. Nonetheless, we resisted the urge to "tweak" the scores by hand.

For each of the surviving files we then took corresponding audio data from a CD, downsampled to mono 8 KHz. Table 2 gives the test set totaling nearly 55 minutes of music. The Mahler example is only the first five or so minutes of the movement. The last movement of the Mozart symphony was broken into two sections due to a repeat that appeared in the MIDI file (and hence our score) but not the performance — a common problem.

For each of these examples we created ground truth by playing the audio file from the computer and tapping along on the keyboard while recording the times of each key press. Each section of constant meter was given a fixed number of taps per measure. These files were then hand-corrected using an interactive program that allows the user to see, hear, and adjust the taps which are superimposed visually over the audio spectrogram and aurally (as clicks) over the sound file. The tap files are not particularly accurate in identifying individual beat locations, but also do not accumulate error over time. That is, they never "get lost."

Our model is completely specified once we describe the model parameters of Eqns. 3 – 6. Of these, the critical parameters are $\{\sigma^2_{s_k}, \sigma^2_{t_k}\}$ since the others only describe the behavior of the initial state and don't seem to have significant effect on the results. We model these variances as parametric functions of the expected note durations, computed using the note lengths $\{l_k\}$ and the local tempo prescribed in the scores derived from the MIDI files. In particular, we model these variances as linear functions of the expected note duration. It would also be possible to model the variances as parametric functions of the expected duration, derived from the expected tempo associated with each branch of the search tree. While we believe these two approaches would give nearly identical results, the latter introduces a modeling complication, since our model assumes the variances are known *a priori*, rather than functions of unknown tempo variables. However, the latter method also uses a more accurate expected duration in the computation of the model variances and might benefit from this. The linear parameters were chosen by trial and error on a training set admittedly not completely distinct from our test set. However, given that we were optimizing only 4 parameters and our data set consists of nearly an hour of music, we expect our results will generalize. While we have made preliminary attempts to learn these parameters from actual data, we have not, as yet, encountered any improvement in performance through this effort.

Errors	Full (HGM)	Restricted (HGM)	Restricted (HMM)
< 1 secs.	.983	.964	.930
$< .5$ secs.	.979	.953	.865
$< .25$ secs.	.951	.916	.607
$< .125$ secs.	.721	.664	.344

Table 3: Percentages of errors under various conditions. Here HGM refers to the Hybrid Graphical Model presented in this paper. HMM refers to a Hidden Markov Model approach, also of our own design. The first column describes performance on the "Full" data set (5038 beats) of Table 2. The next two columns describe the "Restricted" data set (1135 beats) consisting of both examples from *Cosi Fan Tutte* as well as the Shostakovich 1st Symphony Scherzo.

We then processed each of our audio files according to our alignment method, as described in the preceding sections. Every time a branch of the search tree began a note corresponding to a tempo change (as given by the MIDI file), we reset the tempo distribution to have the mean indicated by the MIDI file with a rather large and fixed variance. The result of the process is a collection of estimated onset times, one for each note of the score, written out to a file. For each note that falls on a beat, we compute the "error" as the difference between the estimated onset time and the corresponding tap time. Of the entire collection of 5038 of error estimates (left column Table 3) 95% of the errors were less than

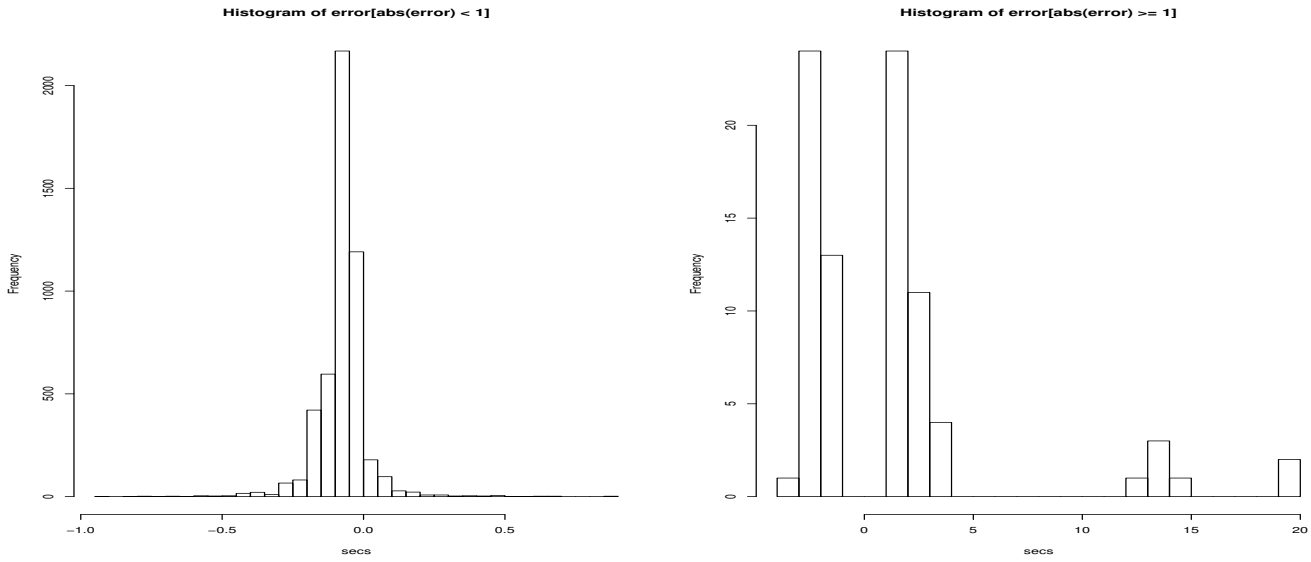

Figure 9: **Left:** Histogram of errors less than 1 sec. in absolute value. **Right:** The remaining errors. Note that similar bar heights represent about 50 times as many counts on the left panel.

.25 sec. in magnitude, while 72% were less than .125 secs. In interpreting these results, one should keep in mind that the tapping ground truth is not especially accurate; thus the measured performance of the algorithm really represents a worse case bound. In fact, our listening of the "click" files for both the recognized data and the ground truth suggests that the recognized results are sometimes more accurate than the ground truth. Since displaying all errors in the same histogram renders the rarer large errors invisible, Figure 9 gives two histograms: the left panel shows the distribution of the errors that are less than 1 sec., while the right histogram gives the remaining errors. Note the 50-fold difference in scale between the two histograms. We suspect that the left histogram really says more about the ground truth than our recognition accuracy.

The left histogram of Figure 9 describes the accuracy of our system when the algorithm is not lost. Figure 10 shows a different aspect of the algorithm's performance by giving the errors, plotted against beat times, for each piece in the collection. In Figure 10 the individual traces are stacked vertically for the sake of comparison, so the errors should be interpreted as deviations from the "baseline," thus we see occasional bursts of errors on the order of several seconds. Figure 10 demonstrates the rather surprising ability of our algorithm to recover after significant errors. In fact, the only places in which our system does not recover from being lost are near the very ends of the excerpts.

The predominant cause of the significant errors appearing in Figure 10 is sections of music with little or no pitch variation. Recall that our data model is based solely on pitch content so the data model contributes essentially no information when the pitch content is static. Not surprisingly, our algorithm experienced difficulty with such sections, as in the repeated *ffff* tritone-laden chords in the brass and strings ending *Mars*; the *pianissimo* open fifths in the strings ending the Shostakovich movement; the harp, string, and timpani *ostinato* that precedes the trumpet trio in *Fêtes*; the long cadence into G Major at the end Mahler excerpt (bar 102); and the final chords of the Mozart overture. This suggests that rather simple improvements to our data model, such as modeling note attacks, might produce better performance in such cases.

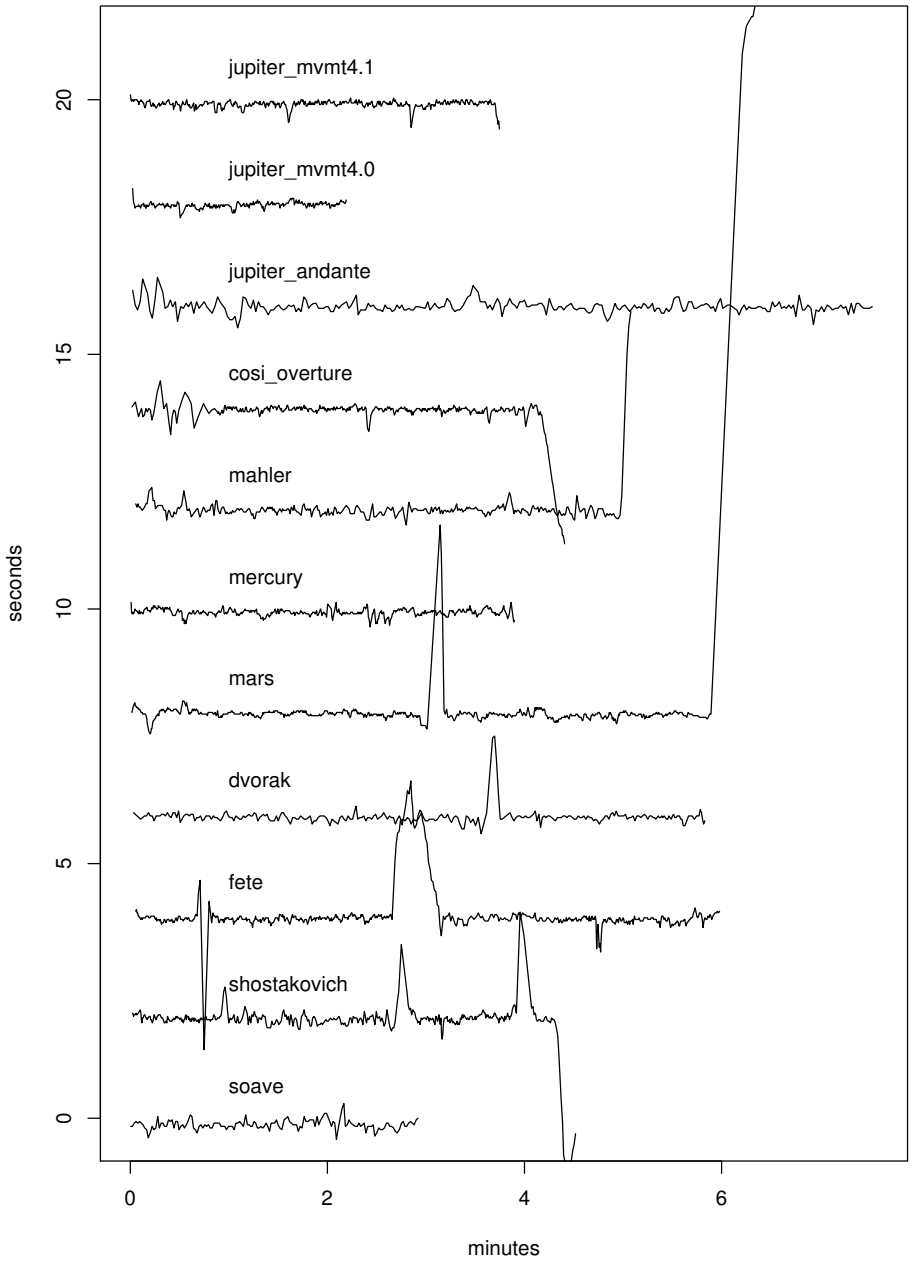

Figure 10: Error vs. beat time for each piece in the data set. The various examples are stacked vertically for ease of comparison, so errors are seen as deviations from the "baseline" rather than actual heights.

The graphs and analyses provide a sense of our algorithm's performance on this difficult test set, however, they are no substitute for an audio demonstration. To this end, we have put "click files" of each piece on the web at

http://xavier.informatics.indiana.edu/~craphael/ismir04

These audio (.wav) files contain the original performance with clicks superimposed at every detected note onset. We encourage the interested reader to listen to some of these examples. In addition, to facilitate comparisons, the above directory also contains the original MIDI files, the tap files, the estimates produced by our algorithm, as well as the original 8KHz audio files.

4.1 Comparison to HMM Approach

As discussed in the introduction, the predominant approach to score alignment has been dynamic-programming-type methods with no simultaneous estimation of tempo. We compared the the method treated here with a method of this type — our HMM system described in [7]. This latter system has been developed with musical accompaniment systems in mind, usually involving a "close-miked" monophonic instrument. While the HMM approach also works for polyphonic instruments such as the piano or guitar, these scenarios represent a significantly different challenge than that of orchestral audio. The most important difference between the HMM approach and the approach of this paper is the treatment of tempo. In the HMM approach, note lengths (really inter-onset intervals) are modeled as indepdent random variables whose parameters are computed from the *global* tempo and printed note length.

We ran our off-line HMM method on a subset of the data described in Table 2. Our HMM method derives its note onset estimates from the posterior distribution of the hidden state variable using the usual forward-backward algorithm. In three of our test examples, *Mercury* from *The Planets*, *Fêtes* from the Debussy *Nocturnes*, and the Mahler 4th Symphony excerpt, our system failed to find the terminal state in the forward pass. Since this is a relatively rare occurance in usual applications of our HMM system, we have not accounted for this, though there are straightforward ways to salvage such a situation. In the remaining three test examples, the Overture and Terzettino from *Cosi Fan Tutte* and the Shostakovich 1st Symphony Scherzo, the forward pass of the HMM approach worked as expected and we computed our usual collection of note onset times. We refer to these last three examples as the "Restricted" test set in Table 3. The last two columns of the table compare the performance of the "hybrid" algorithm of this paper and the HMM approach. These results demonstrate a rather clear superiority of our hybrid approach on this data.

5 Conclusions and Observations

Our motivation for this problem stems from our interest in musical accompaniment systems. One version of our accompaniment system plays back a prerecorded audio file at variable rate (without changing pitch) so as to provide a flexible accompaniment that *follows* a live soloist. In order to acheive synchronization with the soloist we must know where the various audio events occur in both the prerecorded audio and the live audio from the soloist. We have used the algorithm presented herein to generate the match between the prerecorded (usually orchestral) audio and the symbolic score. At present we have performed this operation on around 50 different orchestral movements.

One of the greatest strengths of the proposed method is that it rarely gets "lost." In fact, the experiments presented herein show examples in which the system recovers from interpretations that are locally incorrect, often during regions in which the pitch content alone is insuffiencient to match the score and audio. This ability stems from the our explicit modeling of tempo, allowing the system to have prior expectations which guide our system toward reasonable interpretations, much like a language model in speech. In contrast, the HMM approach proved more brittle in these orchestral experiments.

While the tempo model has clear benefits, it is not without its downside. Situations in which large deviations from the overall tempo occur are often recognized with an overly smoothly varying tempo, sacrificing the precise location of some of the note onsets in the process. Careful review of the results presented on the web will demonstrate this tendency. Many inaccuracies of the method can be traced to an incorrect balance of the timing model and data model — a subject deserving further attention.

In contrast to the score matching method presented here, we have considerable experience using an off-line *hidden Markov model* that does not model tempo, discussed above in Section 4.1. In our HMM system, note lengths are considered as independent variables — an idea criticized in Section 1. This assumption allows the much greater freedom in the position of note onset times and, in practice, puts more weight on the data model. We normally employ our HMM approach with single monophonic instruments or with polyphonic instruments such as piano or guitar. In this environment the HMM system has certain advantages. First of all, it is easy to adapt the HMM to on-line recognition in which the audio data is processed in real-time: in essense, we use the forward probabilities to determine when a note is in the past and then estimate its onset time using the forward-backward algorithm on the available data [7]. Secondly, with the "close-miked" audio data of our accompaniment system, the audio data has more definition and is generally less ambiguous. Thus accurate score matching can be achieved by deemphasizing the timing model and concentrating primarily on finding a match consistent with the audio signal. We still prefer the HMM system for this type of data, though the hybrid model of this paper is suitable for a much wider range of musical scenarios.

6 Acknowledgments

This work supported by NSF grants IIS-0113496 and IIS-0534694. Thanks to the Variations2 Digital Music Library Project at Indiana University for discussions regarding practical aspects of this work.

References

[1] Dannenberg R. "An On-Line Algorithm for Real-Time Accompaniment", *Proceedings of the International Computer Music Conference, 1984* ICMA, Paris, France, 1984.

[2] Vercoe B. "The Synthetic Performer in the Context of Live Performance", *Proceedings of the International Computer Music Conference, 1984* ICMA, Paris, France, 1984.

[3] Baird B., Blevins D., and Zahler N. "Artificial Intelligence and Music: Implementing an Interactive Computer Performer", *"Computer Music Journal* vol. 17, no. 2, 1993.

[4] Puckette, M. "Score following using the sung voice", *Proceedings of the International Computer Music Conference, 1995* ICMA, 1995.

[5] Cont A., Schwarz D. and Schnell N., "Training IRCAM's Score Follower," AAAI Fall Symposium Series on Style and Meaning in Language, Art, Music and Design Proceedings, Washington DC, October 2004.

[6] Grubb L, and Dannenberg R. "A Stochastic Method of Tracking a Vocal Performer", *Proceedings of the International Computer Music Conference, 1997* ICMA, 1997.

[7] Raphael, C. "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models", *IEEE Trans. on PAMI* vol. 21, no. 4, 1999.

[8] Loscos A., Cano P., and Bonada J. "Score-Performance Matching using HMMs", *Proceedings of the International Computer Music Conference, 1999* ICMA, 1999.

[9] Orio, N., and Dechelle, F. "Score Following Using Spectral Analysis and Hidden Markov Models", *Proceedings of the International Computer Music Conference, 2001* ICMA, 2001.

[10] Turetsky R., and Ellis D. "Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI syntheses", *Proc. Int. Symp. Music Info. Retrieval, 2003* Baltimore MD, 2003.

[11] Soulez F., Rodet X., and Schwarz D. "Improving Polyphonic and Poly-Instrumental Music to Score Alignment", *Proc. Int. Symp. Music Info. Retrieval, 2003* Baltimore MD, 2003.

[12] Cemgil, A. T., Kappen, B., "Monte Carlo Methods for Tempo Tracking and Rhythm Quantization," *Journal of Artificial Intelligence* vol. 18, 2003.

[13] Lang, D., de Freitas, N., "Beat Tracking the Graphical Model Way," *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, 2004.

[14] Cemgil, A. T., "Bayesian Music Transcription," *PhD. Thesis* University of Nijmegen, 2004.

[15] Emir K., Pfeffer A., 2005. "Signal-to-Score Music Transcription using Graphical Models", in Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, UK, August 2005.

[16] Raphael, C. "A Hybrid Graphical Model for Rhythmic Parsing", *Artificial Intelligence* vol. 137, no. 1, 2002.

[17] Doucet, A., de Freitas, N., Gordon, N. J. (Eds.) "Sequential Monte Carlo Methods in Practice" *Springer-Verlag, New York*, 2001.