A HYBRID GRAPHICAL MODEL FOR ALIGNING POLYPHONIC AUDIO WITH MUSICAL SCORES

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

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 discrete variables, corresponding to score position, as well as a continuous latent 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 for this task using a variant of dynamic programming involving parametrically represented continuous variables. Experiments are presented on a 55-minute hand-marked orchestral test set.

Keywords: Polyphonic Score Alignment

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. There are two versions of this problem, often called "on-line" and "off-line" recognition or parsing.

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 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 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 user to locate and modify a specific place in a very large audio datafile. 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.

On-line parsing, sometimes called *score-following*, processes the data in real-time as the signal is acquired. Thus, no "look ahead" is possible, as well as imposing speed constraints on the real-time algorithm. 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, however extensions of our approach to on-line parsing are possible.

Many researchers have treated on-line and off-line musical parsing including [2], [12], [1], [6], [3], [7], [4], [5], [9], [11], to name several. See [10] for a thorough bibliography of the subject. 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 and octave slurs, 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 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. 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. 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.

The most straightforward approach to tempo modeling would represent the "state" of the performance as a score position and tempo pair — both discrete variables. 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.

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. The most likely parse is not computable by conventional means, however Section 3 introduces a method by which excellent approximations to the most likely parse can be computed. We attribute the success of the approach to the near-global optimization performed in this section. Section 4 presents results on a 55 minute widely varied test set of short orchestral movements containing examples from Mozart to Shostakovich. The results demonstrate that the note onset estimates are generally quite accurate, and only very rarely does the algorithm become irrecoverably lost. Our dataset has been made publicly available to facilitate comparisons.

2. THE MODEL

In the case of monophonic 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*. In the polyphonic case, the score positions would be created by sorting the collection of all musical event locations (in beats) over all parts, and discarding duplicate positions. Each score position, in beats, would then be associated with the collection of pitches that sound until the next musical event. Thus our simplified score does not represent which notes come from which instruments or distinguish between "attacking" and "sustaining" notes in a chord, although these aspects could be included in a more sophisticated audio model than we use at present. We notate this score by

$$(m_1, c_1), (m_2, c_2), \ldots, (m_K, c_K) \qquad (1)$$

where the kth event begins at m_k beats and c_k is the collection of currently-sounding pitches. 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. 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 model a random process on $t_1, \ldots, t_K$ and $s_1, \ldots, s_K$ through

$$\begin{aligned} s_k &= s_{k-1} + \sigma_k & (2) \\ t_k &= t_{k-1} + l_k s_k + \tau_k & (3) \end{aligned}$$

for $k = 2, \ldots, K$ where l_k is the length, in beats, of the kth chord: $l_k = m_k - m_{k-1}$.

$s_1, \ldots, s_K$ described by Eqn. 2 is our *tempo* process where s_k is the local beat length (secs. per beat) at the kth chord. The σ_k variables are assumed to have 0 mean so the model gives the local tempo at each new musical event as the previous tempo plus a small error. In other words, tempo is modeled as a "random walk."

According to the model, each onset time, t_k, is given as the previous onset time, t_{k-1}, plus the current chord length as predicted by the current tempo ($l_k s_k$), plus a random increment (τ_k). These last random increments, the τ_k, are also assumed to be 0 mean variables so they tend to be small. One possible view of the τ_k variables is as *agogic* accents — at least when they are positive and hence correspond to note lengthenings. However, independent of any musical modeling considerations, these variables give the model a means of explaining note length variations as something other than tempo change, thus stabilizing the model.

The dependency structure of the s and t variables is expressed as a directed acyclic graph in the top of Figure 1. In interpreting this picture, the behavior of each variable (node in the graph) can be described given only its parents in the graph, e.g. t_k depends on t_{k-1} and s_k.

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

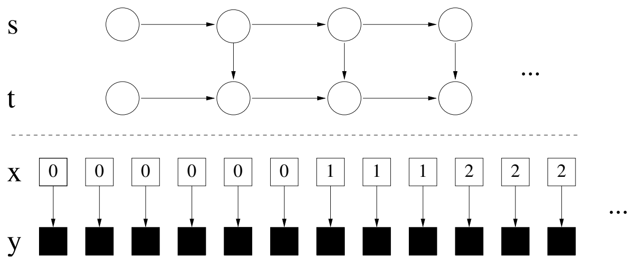

Figure 1. The dependency structure of the model variables expressed as a directed acyclic graph (DAG). Circles denote continuous variables, while squares denote discrete variables. The darkened squares represent observed variables — the spectrogram frames.

ity 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,

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

$k = 2\ldots,K$, where $N(\cdot,\mu,\sigma^2)$ denotes the univariate normal density function with mean μ and variance σ^2. The model parameters $\mu_{s_1},\sigma^2_{s_1},\mu_{t_1},\sigma^2_{t_1}$ and $\{\sigma^2_{s_k},\sigma^2_{t_k}\}_{k=2}^K$ are assumed known. In practice, μ_{s_1} and $\sigma^2_{s_1}$ would be chosen to reflect our knowledge about the initial tempo. The $\{\sigma^2_{s_k},\sigma^2_{t_k}\}_{k=2}^K$ variances should reflect that both tempo changes and note-by-note changes in note length increase with longer note value. The $\sigma^2_{t_1}$ variance should be essentially infinite, corresponding to our lack of knowledge about where the first note of the piece will begin, thereby rendering the choice of μ_{t_1} irrelevant.

Our audio data come from a sampled audio signal which we partition into a sequence of overlapping frames, $y_0,\ldots y_{N-1}$ 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:

$$x_0,x_1,x_2\ldots = \underbrace{00\ldots0}_{I_0}\underbrace{11\ldots1}_{I_1}\underbrace{22\ldots2}_{I_2}\ldots \quad (6)$$

would signify that the first note begins at $I_0\Delta$ seconds and lasts for $I_1\Delta$ seconds, the second note begins at $(I_0+I_1)\Delta$ seconds and lasts for $I_2\Delta$ seconds, etc. In our model each frame of audio data depends only on the current chord which is sounding, thus if $x = x_0,\ldots,x_{N-1}$ and $y = y_0,\ldots,y_{N-1}$, we have

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

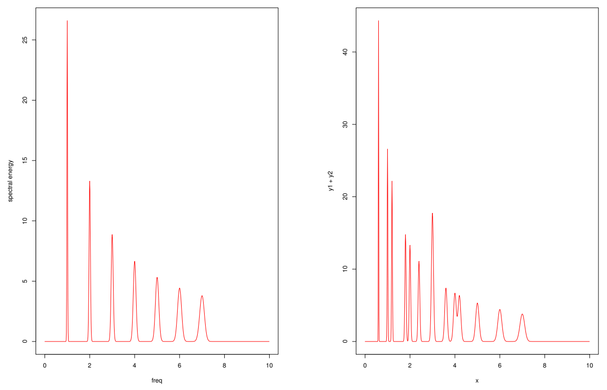

Figure 2. **Left:** An idealized spectrum for a single note. **Right:** An analogous spectrum for two simultaneous notes created as a superposition of two single-note spectra.

The *conditional* distribution of y given x is depicted in the bottom of Figure 1.

Our actual data model is given by

$$p(y_n|x_n) = c(y_n)\prod_{f=1}^{F} g_{x_n}(f)^{y_n(f)/\alpha} \quad (7)$$

In Eqn. 7, g_{x_n} is an idealized power spectrum associated with chord indexed by x_n and is obtained as a superposition of the idealized individual note spectra and then normalized to sum to 1 over the frequency variable, f. The left panel of Figure 2 shows an example of an idealized single-note spectrum, while the right panel gives the idealized spectrum for a two-note chord. Thus, the right spectrum, normalized to sum to one would play the role of g_{x_n} in Eqn. 7 for the corresponding two-note chord. in Eqn. 7, y_n is the observed spectrum for the nth audio frame, and α is a scaling constant that weights the contribution of the data term to the overall model. The factor $c(y_n)$ can be disregarded since the y_n variables are fixed, thus the $c(y_n)$'s are constant. This data model would result if we were to assume that the observed spectrogram was the histogram of a random sample from the probability distribution given by g_{x_n}. However, even without such justification, the model is intuitively plausible.

We are interested in specifying a *joint* model on the variables 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 lose nothing by eliminating t and modeling $p(s,x,y)$. To do this, note that any sequence $x = x_0,\ldots,x_{N-1}$ implicitly fixes the the actual onset times through

$$t_k(x) = \min\{n : x_n = k\}\Delta \quad (8)$$

so we have $p(s,x) = p(s,t(x))$ where the later probability is given by the model of Eqns. 2 and 3. We have

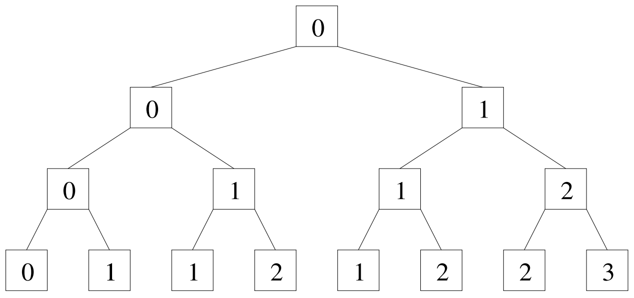

Figure 3. The search tree associated with optimization. The label of a tree node is the score index describing the current chord.

already described the conditional distribution $p(y|x)$, so

$$\begin{aligned} p(s,x,y) &= p(s,x)p(y|x) \quad (9) \\ &= p(s_1)p(t_1) \\ &\times \prod_{k=2}^{K} p(s_k|s_{k-1})p(t_k|t_{k-1},s_k) \quad (10) \\ &\times \prod_{n=0}^{N-1} p(y_n|x_n) \end{aligned}$$

While the right hand side of Eqn. 10 appears to depend on t, we recall that $t_k = t_k(x)$, as in Eqn. 8.

We cannot write any simple DAG representation of the probability distribution on s, x, y. The actual DAG would be completely connected between the s and x layers. However, the model is still computationally tractable, as we show in the following section.

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:

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

If all variables were discrete, this problem would be solvable through traditional dynamic programming techniques, however, note that the tempo process, s, is continuous. We describe here a method for approximating the global solution to Eqn. 11 using a technique that is similar to dynamic programming, but allows us to treat continuous variables.

Consider the tree of Figure 3, 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 $x_0,\ldots,x_n$, and hence an alignment of the first $n+1$ frames of audio data; clearly all possible realizations are contained in the tree.

Traversing a partial path through the tree (fixing $x_0,\ldots,x_n$) implies that the first $n+1$ frames contain k notes where $k = x_n$. Additionally, the first variables $t_1,\ldots,t_k$ are also determined through Eqn. 8. For the partial path, $x_0,\ldots,x_n$, 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

$$\begin{aligned} p(s_1^k,x_0^n,y_0^n) &= p(s_1)p(t_1) \\ &\times \prod_{\kappa=2}^{k} p(s_\kappa|s_{\kappa-1})p(t_\kappa|t_{\kappa-1},s_\kappa) \quad (12) \\ &\times \prod_{\nu=0}^{n} p(y_\nu|x_\nu) \end{aligned}$$

(again $t_k = t_k(x_0^n)$) For each partial path x_0^n define

$$\hat{p}_{x_0^n}(s_k) = \max_{s_1\ldots,s_{k-1}} p(s_1^k,x_0^n,y_0^n) \quad (13)$$

$\hat{p}_{x_0^n}(s_k)$ gives the quality of a particular path x_0^n as a function of the current tempo s_k, assuming the most favorable choice of past tempo variables $s_1,\ldots,s_{k-1}$.

While the calculations are somewhat involved, $\hat{p}_{x_0^n}(s_k)$ can be shown to follow the simple parametric form

$$K(s;h,m,v) = he^{-\frac{1}{2}(s-m)^2/v} \quad (14)$$

with parameters h, m, v. In fact, a simple recursion can be developed for this function as one proceeds down a particular path in the tree of Figure 3, as follows. Suppose $x_0^{n-1} = (x_0,\ldots,x_{n-1})$ is such that $x_{n-1} = k$ and $\hat{p}_{x_0^{n-1}}(s_k) = K(s_k;h,m,v)$. There are two cases. First, if $x_n = x_{n-1}$ (the kth chord persists through the nth frame), then

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

Otherwise,

$$\begin{aligned} \hat{p}_{x_0^n}(s_{k+1}) &= \max_{s_k} \hat{p}_{x_0^{n-1}}(s_k)p(s_{k+1}|s_k) \\ &\times p(t_{k+1}|t_k,s_{k+1})p(y_n|x_n) \quad (15) \\ &= K(s_{k+1};h',m',v') \end{aligned}$$

where

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

Note that, rather than having a single "score" for each partial path, x_0^n, we describe the quality of the path *as a function of the current tempo*, s_k: $\hat{p}_{x_0^n}(s_k) = K(s_k;h,m,v)$. In interpreting this representation, the scaling parameter,

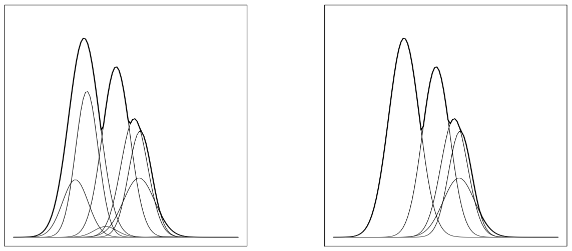

Figure 4. **Left:** The functions $\{\hat{p}_{x_0^n}(s_k)\}$ before **Right:** The reduced collection of functions after thinning.

h, gives an overall description of the quality of the partial path since it is the maximal probability attainable. That is

$$\max_s K(s;h,m,v) = K(0,h,m,v) = h$$

m describes the best value of the current tempo, s_k, for the path. v is a measure of how fast the path quality decreases as we move away from the best tempo, m.

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 that some partial paths can be eliminated without sacrificing the search for the optimal path.

Suppose that two partial paths x_0^n and $\tilde{x}_0^n$ are such that both begin the kth note at the nth frame: $x_n = \tilde{x}_n = k$ and $x_{n-1} = \tilde{x}_{n-1} = k-1$. If we also have $\hat{p}_{x_0^n}(s_k) > \hat{p}_{\tilde{x}_0^n}(s_k)$ for all values of the current tempo s_k, then no matter how the paths continue, the x_0^n branch will always beat the $\tilde{x}_0^n$ branch. So, without any loss, we can eliminate the latter branch.

More generally, suppose that I is the collection of paths that begin the kth note on the nth frame, i.e.

$$I = \{x_0^n : x_{n-1} = k-1, x_n = k\}$$

Define the set $\mathrm{Thin}(I)$ as the smallest subset of I such that

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

for all s_k as in Figure 4. Thus $\mathrm{Thin}(I)$ are the Gaussians that attain the maximum value for some tempo value. Paths not in I are not optimal for any value of s_k, so, reasoning as above, we can eliminate any such path 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 $|I|$. The thinning algorithm can be performed on a restricted set of possible tempo values, say $(s_{\min}, s_{\max})$ to achieve greater reduction of the partial paths. An algorithm for the thinning operation is discussed in [8] as well a discussion of computational complexity and the optimality properties of the restricted thinning algorithm.

In our experiments the number of kernels that survive the thinning process does not increase with the number of original kernels. Assuming that the number of surviving kernels of each thinning operation is bounded by some maximum, the thinning procedure reduces the number of partial paths at each frame to a number that is, at worst, linear in K — the number of chords in the score. The paths surviving thinning represent a tiny fraction of what would exist otherwise since, without thinning, the number of partial paths grows exponentially in n, the current analysis frame. In our applications, K can be in the thousands, so an algorithm that is linear in K is still not tractable and more pruning must be done. We discuss further pruning in the following section.

The final parse is obtained by tracing back the best surviving path at the final frame, N. That is, our parse estimate is $\hat{x}_0^N = \arg\max_{x_0^N} h(x_0^N)$ where

$$\hat{p}_{x_0^N} = K(s_K; h(x_0^N), m(x_0^N), v(x_0^N))$$

and only paths that reach the final score position, K, in the Nth frame are considered. It is also straightforward to recover the hidden tempo variables, although we do not do so in this particular application.

3.1. Further Pruning

As observed above, we still need to prune paths to make the proposed algorithm feasible. A simple approach would be to sort the current (surviving) hypotheses on the "h" parameter and prune the smallest of these. 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. 12. Since k varies over the partial paths, the h-scores are composed of different numbers of factors and direct comparison is suspect.

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

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

In the above equation, $D(x)$ is simply the data log likelihood of the partial path. $M(x)$ 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, implicitly fixes the first k position variables $t_1, \dots, t_k$, we only maximize over the remaining variables. While we omit the calculation, one can easily compute $M(x)$ recursively as n increases. At any fixed iteration, then we are sorting over the data log likelihood plus a constant times the *average* model likelihood, therefore not penalizing the paths with more notes. However, as $n \to N$ and $k \to K$ this gradually reduces to the the original log likelihood $D(x) + M(x)$ as in 13.

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 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 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.

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*. 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 1 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.

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	Berstein	1989	4.88

Table 1. The test set for the score matching experiments.

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 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 variances of Eqns. 4 and 5, $\{\sigma^2_{s_k}, \sigma^2_{t_k}\}$. We model the variances as parametric functions of the expected note durations, computed using the note lenths $\{l_k\}$ 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 brach 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 unkwnown 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, though we will estimate a more complex parametric model in future work.

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 vari-

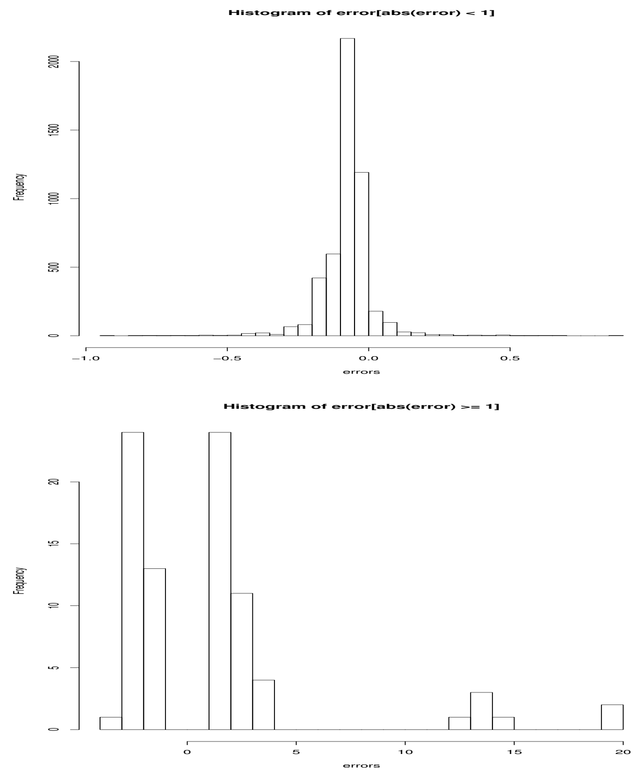

Figure 5. **Top:** Histogram of errors less than 1 sec. in absolute value. **Bottom:** The remaining errors. Note that similar bar heights represent about 50 times as many counts on the left panel.

ance. 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 error estimates, 95% of the errors were less than .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, thereby making the measured performance of the algorithm appear worse than they are in truth. In fact, our listening of the "click" files for both the recognized data and the ground truth suggests that the recognized results are generally more accurate than the ground truth. Since displaying all errors in the same histogram renders the rarer large errors invisible, Figure 5 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 histograms of Figure 5 show that our algorithm gives accurate performance when it is not lost. Figure 6 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 6 the individual traces

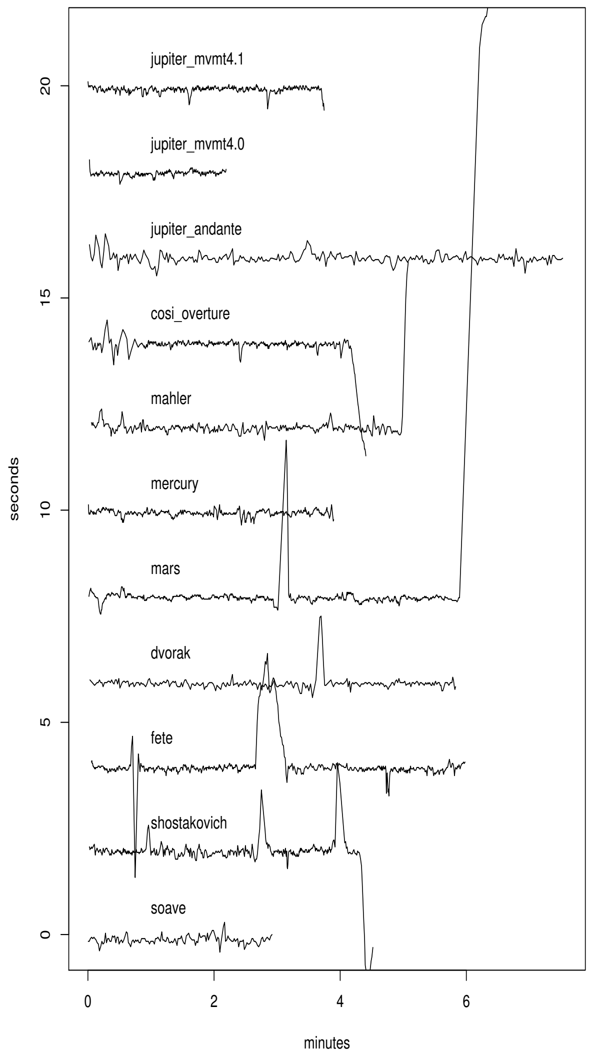

Figure 6. Error vs. beat time for each piece in the data set. The various examples are stacked vertically for ease of comparsion, so errros are seen as deviations from the "baseline" rather actual heights.

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 6 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 6 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* the 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.

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://fafner.math.umass.edu/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.

5. ACKNOWLEDGMENTS

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

6. REFERENCES

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

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

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

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

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

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

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

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

[9] 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.

[10] Schwarz D. "Score Following Commented Bibliography", http://www.ircam.fr/equipes/temps-reel/suivi/bibliography.html IRCAM, 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] Vercoe B. "The Synthetic Performer in the Context of Live Performance", *Proceedings of the International Computer Music Conference, 1984* ICMA, Paris, France, 1984.