SCORE FOLLOWING WITH HIDDEN TEMPO USING A SWITCHING STATE-SPACE MODEL

Yucong Jiang
University of Richmond
yjiang3@richmond.edu

Christopher Raphael
Indiana University Bloomington
craphael@indiana.edu

ABSTRACT

A score-following program traces the notes in a musical score during a performance. This capability is essential to many meaningful applications that synchronize audio with a score in an on-line fashion. Existing algorithms often stumble on certain difficult cases, one of which is piano music. This paper presents a new method to tackle such cases. The method treats tempo as a variable rather than a constant (with constraints), allowing the program to adapt to live performance variations. This is first expressed by a Kalman filter model at the note level, and then by an almost equivalent switching state-space model at the audio frame level. The latter contains both discrete and continuous hidden variables, and is computationally intractable. Weshow how certain reasonable approximations make the computation manageable. This new method is tested on a dataset of 50 piano excerpts. Compared with a previously established state-of-the-art algorithm, the new method shows more stable and accurate results: it reduces fatal score-following errors, and improves accuracy from 65.0% to 69.1%.

1. INTRODUCTION

The score-following problem involves building a computer program that can trace musical events in a given musical score during a live performance. This is called "online audio-to-score alignment", which constantly figures out the current position in the score while the performance is going on; the program can only access the audio received before the current moment. In contrast, *off*line audio-to-score alignments start the task *after* the performance is done, allowing the program to access the entire recording.

Score following enables a number of useful applications: a musical score page turner [1], automatic accompaniment systems [2], virtual scores designed to react to a live performance [3], real-time audio enhancement during a music performance, and even a computer tutor [4].

A typical score-following algorithm infers the score positions by evaluating the hypothesis paths through a state graph, each path representing a possible performance. From the Bayesian point of view, the evaluation criteria for a path include two aspects: how much a hypothesis is consistent with the *prior model*, which represents the anticipation of the musical performance *before* observing any data, and how much the data support a hypothesis, which is called the *data model*. This paper focuses on improving the former aspect, which is essentially about timing—when a note appears and how long it lasts.

Many researchers have considered timing in their prior models. [5, 6] used a hidden semi-Markov model where the duration of each state represents a note length. Others chose to directly model the tempo as a latent variable: [7, 8] treated the tempo as a discrete variable, while [9–12] adopted continuous state-space models, and used particle filter to approximate the results. [13] developed a hybrid graphical model, and tested it by aligning orchestra music offline. [13] provides a jumping-off point for the proposed method here. Note that feature-based methods (e.g., DTW or DNN) are beyond the scope of this discussion.

Unfortunately, existing score-following algorithms can still stumble on some challenging cases, especially when the data model is not reliable; e.g., shared notes among neighboring chords, extended sound from previous chords by pedaling, and blurring effects caused by fast playing, as in piano music. This paper presents a new method aiming to improve the timing model—this aspect is especially meaningful in those challenging cases. In practice, we can assume that the tempo tends to be smooth: the tempo is steady most of the time, sometimes floating around slowly, but rarely jumps up and down abruptly. Therefore, the new method models the tempo as a continuous variable, and it is smooth. This allows the note lengths (or the local tempo) to adapt to the performance data. One of the most popular state-space models, the Kalman filter model, is suitable for tracking such a tempo variable (Section 3.1). It becomes a *switching* state-space model after changing the scope of time (Section 3.2). Section 4 shows how the increased computational complexity is manageable with approximations. This new method was tested on real piano performance data presented in Section 5.

2. REPRESENTING SCORE AND AUDIO

We can view the musical score in a "homophonic" way, representing the score as a sequence of chords, as in Figure 1. It enables polyphonic music to be linearly represented as the same fashion as in monophonic music—a sequence of chords, each chord associated with a score position.

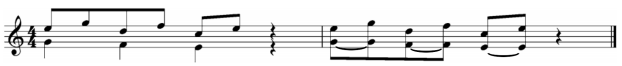

Figure 1. "Homophonic" view of polyphonic music [14]. The left bar is the original score with two voices. The right bar is its "homophonic" view.

The audio is sampled, and evenly segmented into frames, with some overlap between adjacent frames. Each frame is transformed into a Fourier spectrum [15]. The *data model* here defines the likelihood of observing the data spectrum given the chord index. Denote y as the spectrum of a frame, a vector $\{y_w\}$, $1 \leq w \leq W$. Denote the template (see [13]) of the kth chord as $q^k = \{q_w^k\}$, which sums to 1. The likelihood of observing the data y given the index k is:

$$p(y|k) = \prod_{w=1}^{W} (q_w^k)^{y'_w} \tag{1}$$

where $y'_w = y_w / \sum_{w=1}^{W} y_w$.

3. THE MODEL

3.1 Kalman Filter Model for Tempo

The polyphonic score is represented as a sequence of chords, with a new chord appearing whenever any note is added, ended, or changed in the current chord. Let's assume there are K such chords in the musical score, each chord with a nominal musical length (e.g., 1/4 for a quarter note and 1 for a whole note) represented by l_k, $k = 1, \ldots, K$. Weassume every chord has its own tempo value, and it doesn't change within a chord's lifetime. Let t_k be the tempo of the kth chord (seconds per whole note), and o_k be this chord's onset time (in seconds). The joint evolution of the tempo and the onset is modeled as a linear dynamical system. If we treat the onsets as observable data, the formula is the same as a Kalman filter model:

$$o_{k+1} = o_k + l_k t_k + \varepsilon_{k+1} \tag{2}$$

$$t_{k+1} = t_k + \eta_{k+1} \tag{3}$$

where all random variables have normal distributions:

$$\begin{aligned} o_1 &\sim N(\mu_{o,1}, \sigma_{o,1}^2) \\ t_1 &\sim N(\mu_{t,1}, \sigma_{t,1}^2) \\ \varepsilon_k &\sim N(0, \sigma_{\varepsilon,k}^2),\ k = 2, \ldots, K \\ \eta_k &\sim N(0, \sigma_{\eta,k}^2),\ k = 2, \ldots, K \end{aligned}$$

The ε_k's and η_k's are all mutually independent, and they are independent from o_1 and t_1 as well. (In the experiments, however, $\sigma_{\varepsilon,k}$ was modified to be proportional to t_k, and $\sigma_{\eta,k}$ proportional to $l_k t_k$, with manually set scales.) The dependency graph is in Figure 2. In the rest of this paper, we refer to a "chord" as a "note" for simplicity's sake.

3.1.1 Marginal Likelihood of Onsets

This linear dynamical system can be viewed as a Markov process of generating the onsets, $o_1^K = (o_1, o_2, \ldots, o_K)$,

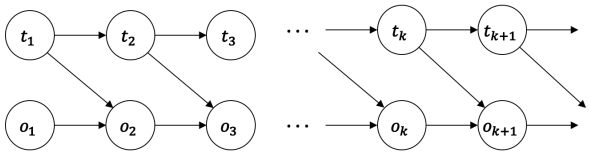

Figure 2. Dependency graph for the tempo and the onset.

demonstrated as follows. According to the chain rule and the described model, we can write

$$\begin{aligned} p(o_1^K) &= p(o_1) \prod_{k=1}^{K-1} p(o_{k+1}|o_1^k) \\ &= p(o_1) \prod_{k=1}^{K-1} \int_{t_k} p(o_{k+1}|t_k, o_k) p(t_k|o_1^k)\, \mathrm{d}t_k \end{aligned}$$

Because $o_{k+1} = o_k + l_k t_k + \varepsilon_{k+1}$, the integral factors can be further simplified as

$$\begin{aligned} &\int_{t_k} N(o_{k+1}; o_k + l_k t_k, \sigma_{\varepsilon,k+1}^2) N(t_k; \mu_t(o_1^k), \sigma_t^2(o_1^k))\, \mathrm{d}t_k \\ &= N(o_{k+1}; o_k + l_k \mu_t(o_1^k), \sigma_{\varepsilon,k+1}^2 + l_k^2 \sigma_t^2(o_1^k)) \end{aligned}$$

where $\mu_t(o_1^k) = E(t_k|o_1^k)$ and $\sigma_t^2(o_1^k) = Var(t_k|o_1^k)$, which can be iteratively calculated using a Kalman filter as the system receives $o_1, o_2, \ldots, o_k$ [16]. Thus, we have

$$\begin{aligned} &p(o_1^K) = \\ &p(o_1) \prod_{k=1}^{K-1} N(o_{k+1}; o_k + l_k \mu_t(o_1^k), \sigma_{\varepsilon,k+1}^2 + l_k^2 \sigma_t^2(o_1^k)) \end{aligned} \tag{4}$$

which can be computed iteratively as k increases.

3.2 Frame-wise Representation

The discussions in Section 3.1 is based on the linear dynamical system at the note level, as in Figure 2 where the discretized "time step" is the note index. However, in real-world applications, the audio is received frame by frame (every 16 milliseconds in the experiments). In order to incorporate such audio frames as observed data, we have to change the scope and view the model in Figure 2 at the *frame* level, with each audio frame as a time step. Let's use n to denote the index of a frame, $n = 1, \ldots, N$, where N is the total number of frames in the audio. Any frame, n, has a label variable, $k_n \in \{0, \ldots, K\}$, which is the index of the sounding note at that frame. Frames before the first note being played are labeled as 0, i.e., $\{n : k_n = 0\}$. Denote y_n as the observed audio at the nth frame. We can assume the distribution of the audio frame data only depends on this frame's label—which note is being played. Figure 3 shows the model at both levels in the same graph.

The note onsets and the frame labels are nearly interchangeable. Let Δ be the time difference between adjacent frames (in milliseconds), a sequence of frame labels, k_1^n, can be recovered from a sequence of onsets, o_1^k, or vice versa, like this:

$$k_n = \min\left\{k \in \{0, \ldots, K\} :\ n\Delta < o_{k+1}\right\} \tag{5}$$

$$o_k \approx \Delta \cdot \min\left\{n \in \{1, \ldots, N\} :\ k_n = k\right\} \tag{6}$$

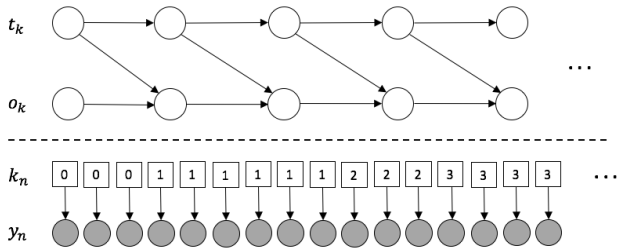

Figure 3. Dependency graph. Upper panel is note level; lower panel is frame level. Circles are continuous variables; squares are discrete. Observed variables are shaded.

The note-wise representation and the frame-wise representation are almost equivalent, except that there are two additional assumptions in the latter. First, the note onsets are now discrete because they have to be multiples of Δ, as in Equation 6. Second, a note must last at least one frame long, so $o_{k+1} - o_k \geq \Delta$. Because the labels and the onsets can be derived deterministically from each other as in the above two equations, the onsets can actually be viewed as "discrete" variables, and the tempo variables are the only real continuous (hidden) variables. Such state-space models that involve both discrete and continuous hidden variables are called *switching* state-space models [17].

3.2.1 Marginal Likelihood of Labels

This section describes a generative model for the frame labels, k_1^N, (almost) equivalent to the generative model in Section 3.1.1, but at the frame level. Using the chain rule, we have

$$p(k_1^N) = p(k_1) \prod_{n=1}^{N-1} p(k_{n+1}|k_1^n) \tag{7}$$

Following the assumptions in the frame-wise representation, there could be only two possible values for k_{n+1} in each factor $p(k_{n+1}|k_1^n)$: the same as k_n if it "decides" to stay at the current note, or $k_n + 1$ if it "decides" to move on to the next note. From Equation 4, we know that the onset of the pending note $k_n + 1$, given all previous onsets $o_1^{k_n}$ (equivalent to k_1^n), has a density function

$$p(o_{k_n+1}|o_1^{k_n}) = \tag{8}$$
$$N(o_{k_n+1}; o_{k_n} + l_{k_n}\mu_t(o_1^{k_n}), \sigma^2_{\varepsilon,k_n+1} + l^2_{k_n}\sigma^2_t(o_1^{k_n}))$$

Let's write ϕ as the standard normal density, and define

$$f(x) = \phi(\frac{x-\mu}{\sigma})$$

Then, the density in Equation 8 is $f(o_{k_n+1})$, where

$$\mu = o_{k_n} + l_{k_n}\mu_t(o_1^{k_n})$$
$$\sigma = \sqrt{\sigma^2_{\varepsilon,k_n+1} + l^2_{k_n}\sigma^2_t(o_1^{k_n})}$$

We can use this density function to compute $p(k_{n+1}|k_1^n)$ with two cases:

$$p(k_{n+1}|k_1^n) = \begin{cases} p_1, & k_{n+1} = k_n \ (same\ note) \\ 1 - p_1, & k_{n+1} = k_n + 1 \ (new\ note) \end{cases} \tag{9}$$

We can focus on calculating the first case, and the second case has the complimentary probability. In the first case where it stays in the same note at frame $n+1$, the information we know is that the onset of the next note will be *after* frame $n+1$, given we already know that the onset must be after frame n. Therefore, p_1 is a conditional probability:

$$p_1 = P(o_{k_n+1} > (n+1)\Delta \mid o_{k_n+1} > n\Delta\,, o_1^{k_n}) \tag{10}$$

Writing Φ as the cumulative distribution function of ϕ, we have

$$p_1 = \frac{1 - \Phi(\frac{c+\Delta-\mu}{\sigma})}{1 - \Phi(\frac{c-\mu}{\sigma})} \tag{11}$$

where $c = (n+1)\Delta$. Therefore, we can calculate the probability of any label sequence $p(k_1^N)$ iteratively as in Equation 7, by using Equation 9.

3.2.2 Note Duration and Note Age

In Equation 9, $p(k_{n+1}|k_1^n)$ is calculated based on $f(o_{k_n+1})$, the distribution of the onset for the pending note. In this section, we introduce two variables—note duration and note age—and calculate $p(k_{n+1}|k_1^n)$ from a slightly different perspective.

The duration of the kth note is the difference between its two adjacent onsets:

$$L_k = o_{k+1} - o_k = l_k t_k + \varepsilon_{k+1}$$

Given the previous onsets, L_k also has a Gaussian distribution with its mean as $l_k\mu_t(o_1^k)$ and its variance as $\sigma^2_{\varepsilon,k+1} + l^2_k\sigma^2_t(o_1^k)$. In the frame-wise representation, a note's duration, L_{k_n}, has additional requirements that it should be multiples of Δ, and be at least Δ (milliseconds) long. Let's define a note's age as the number of frames this note has been through so far at the nth frame, denoted as a_n. The age of the note k_n can be calculated by

$$a_n = n - o_{k_n}/\Delta + 1$$

To get p_1 in Equation 9, we can rewite Equation 11 from the angle of the note length: given that this note has lasted a_n frames, what's the probability of it lasting for at least $a_n + 1$ frames. It can be expressed as

$$\begin{aligned} p_1 &= P(L_{k_n} \geq (a_n+1)\Delta \mid L_{k_n} \geq a_n\Delta\,, o_1^{k_n}) \\ &= \frac{1 - \Phi(\frac{(a_n+1)\Delta-\mu}{\sigma})}{1 - \Phi(\frac{a_n\Delta-\mu}{\sigma})} \end{aligned} \tag{12}$$

where $\mu = l_{k_n}\mu_t(o_1^{k_n})$ and $\sigma = \sqrt{\sigma^2_{\varepsilon,k_n+1} + l^2_{k_n}\sigma^2_t(o_1^{k_n})}$.

4. COMPUTATION

The number of possible label sequences grows exponentially with n, as in the tree structure in Figure 4. This section discusses the computational aspects of the model: what is the filtered probability of the hidden variables, given all observed audio data up to the current frame; how to reasonably approximate the calculation so it is tractable.

4.1 Tree Representation

The tree in Figure 4 represents all possible label sequences, $\{k_1^N\}$. At any frame n, each node has a label for its note index, k_n, and also includes the age information of the note—the number of frames it has been through so far—denoted by a_n. From a label sequence k_1^n, we can thus determine the age sequence a_1^n, and vice versa. The tree includes the age variable because we need it in the generative model (see Equation 12). A node has two children: left means it stays in the same note and thus the age increases by one frame, and right means it moves on to the next note and thus the age resets to 1. Any node in the tree actually represents a label sequence by the path from the root to the node. For example, the dotted line in Figure 4 represents the label sequence of $k_1^6 = 011122$ (or the age sequence of $a_1^6 = 112312$). A path can be viewed as a sequence of decisions of choosing the left or right branch, from the root node at frame 1 to the end node in the path.

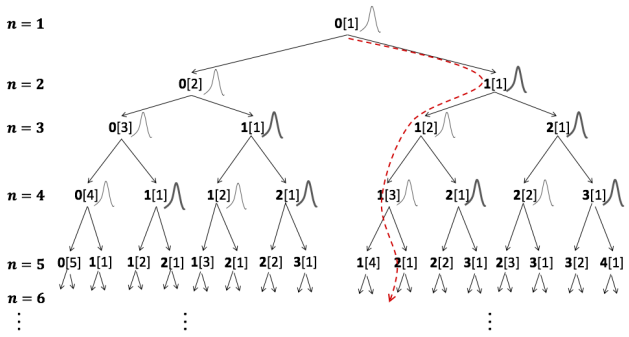

Figure 4. Exponential growth of label sequences. Each node has three aspects: note index label, age of this note (in square brackets), and distribution of the tempo. Tempo distributions at *onset* nodes are drawn with thicker lines.

For every node in the tree, denoted by its path of k_1^n, we can calculate two probabilities: the discrete probability of arriving at this node, $p(k_1^n)$, and the continuous probability distribution of the tempo at this node, $p(t_{k_n}|k_1^n)$. As discussed before, the tempo has a Gaussian distribution (drawn besides the nodes in the first four frames in Figure 4), given the path leading to this node. A Kalman filter keeps track of the tempo down the path, and updates its distribution when and only when the path chooses a right branch (drawn with thicker lines), i.e., whenever it observes a note onset (discussed in Section 3.1).

The probability of arriving at a node, $p(k_1^n)$, according to the frame-wise generative model, can be iteratively calculated by $p(k_1^n) = p(k_1^{n-1})p(k_n|k_1^{n-1})$. Thus, $p(k_1^n)$ can be calculated from two parts: the probability of arriving at its parent node, $p(k_1^{n-1})$, and the probability of choosing the left or the right branch when transitioning from frame $n-1$ to frame n, $p(k_n|k_1^{n-1})$. We can use either Equation 11 or Equation 12 to calculate the latter, and both equations require the distribution of the tempo at the parent node, $p(t_{k_{n-1}}|k_1^{n-1})$—or equivalently, $p(t_{k_{n-1}}|o_1^{k_{n-1}})$.

Adopting the iterative nature of the calculation, we can compute the probability of (arriving at) every node and its tempo distribution, frame by frame starting from the root. Since the tree includes every possible label sequence, those probabilities give us $p(k_1^n)$ for all $\{k_1^n\}$, $n = 1, \ldots, N$.

4.2 Conditioning on data

This section continues to focus on calculating the probability of arriving at a node in the tree, but now *conditioned* on the observed audio data up to the current frame—$p(k_1^n|y_1^n)$. Considering y_1^n ensures that sequences more consistent with the observed data would receive higher probabilities than the rest, helping identify more likely sequences. On the other hand, the tempo will not be affected by the observed data since the tempo distribution at a node depends only on the corresponding label sequence, i.e., $p(t_{k_n}|k_1^n, y_1^n) = p(t_{k_n}|k_1^n)$, as used later in Equation 13.

The audio frame data $y_1, \ldots, y_n$ are assumed to be conditional independent from each other given the frame labels $k_1, \ldots, k_n$, so we can have

$$\begin{aligned} p(k_1^n|y_1^n) &= \frac{1}{Z_n} p(k_1^n, y_1^n) \\ &= \frac{1}{Z_n} p(k_1^n) p(y_1^n|k_1^n) \\ &= \frac{1}{Z_n} p(k_1^n) \prod_{i=1}^{n} p(y_i|k_i) \\ Z_n = p(y_1^n) &= \sum_{\{k_1^n\}} p(k_1^n, y_1^n) \end{aligned}$$

The joint probability $p(k_1^n, y_1^n)$ can be calculated iteratively from $p(k_1^{n-1}, y_1^{n-1})$ by

$$p(k_1^n, y_1^n) = p(k_1^{n-1}, y_1^{n-1}) p(k_n|k_1^{n-1}) p(y_n|k_n)$$

The calculation of the middle factor $p(k_n|k_1^{n-1})$ is discussed in Equations 9, 11, and 12, which involve using the tempo distribution of $p(t_{k_{n-1}}|k_1^{n-1})$. The third factor $p(y_n|k_n)$ is the data likelihood of the nth frame, discussed in Section 2. Therefore, we can calculate $p(k_1^n, y_1^n)$ for every node in the tree iteratively from frame 1 to frame N. To get the filtered probability of $p(k_1^n|y_1^n)$, we simply normalize $p(k_1^n, y_1^n)$ with Z_n at every frame. In sum, with the help of the data model factors $\prod_{i=1}^{n} p(y_i|k_i)$, we should be able to distinguish the sequences better by using $p(k_1^n|y_1^n)$ instead of $p(k_1^n)$: hypothesized sequences that are closer to the true sequence should have larger values of $p(k_1^n|y_1^n)$ than those of sequences further from the truth.

4.3 Filtering with Approximation

The task of filtering is to compute $p(k_n, t_{k_n}|y_1^n)$: the joint distribution of the last hidden states in the sequence, given the sequence of observed data so far. Writing $\mathcal{K}(k_n)$ as the set of all label sequences in Figure 4 that are n-label long and end with k_n, this filtered probability is:

$$\begin{aligned} p(k_n, t_{k_n}|y_1^n) &= \sum_{k_1^n \in \mathcal{K}(k_n)} p(k_1^n, t_{k_n}|y_1^n) \\ &= \sum_{k_1^n \in \mathcal{K}(k_n)} p(k_1^n|y_1^n) p(t_{k_n}|k_1^n) \end{aligned} \quad (13)$$

This is a Gaussian mixture distribution with $|\mathcal{K}(k_n)|$ components. Without approximation, the computation is intractable because $|\mathcal{K}(k_n)|$ grows exponentially with n.

In Figure 4, there are duplicated nodes in terms of "label[age]" at every level (except for the first three levels), and these duplicates have the same subtree. The idea for simplifying the computation is to merge any duplicates at the same level into one node, as in Figure 5. The merged node then has the summed probability:

$$p(\text{merged node}) = \sum\nolimits_{\text{nodes}} p(\text{node}_i)$$

Each node in the frame-wise model also carries a Gaussian distribution for the tempo. The merged node would then carry a Gaussian mixture distribution:

$$p(t_{\text{merged}}) = \sum\nolimits_{\text{nodes}} \frac{p(\text{node}_i)}{p(\text{merged node})} p(t \mid \text{node}_i)$$

As more and more merging operations happen over time, the number of components in a tempo distribution grows exponentially, with each component corresponding with a possible label sequence, and the problem remains intractable. We can solve this by using *moment matching*: approximating a Gaussian mixture with a single Gaussian that has the same mean and variance as the mixture. This is proven to be the optimal approximation in the sense of Kullback-Leibler distance [18]. This approximation is reasonable if the components with larger weights are close to each other (agreeing with each other) in terms of mean and variance, and if those further away have smaller weights, thus could be ignored anyway. It's reasonable to believe that reality more often reflects this case, because unlikely nodes tend to have smaller probabilities and thus smaller weights in the mixture (thus should be ignored anyway), while more likely nodes tend to have more similar opinions about the tempo because they lean towards the truth.

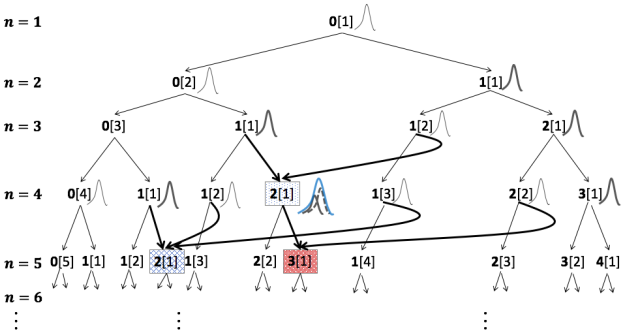

Figure 5. Limiting tree growth by merging nodes with the same label and age. The first merge happens at frame #4, when the right child of 1[1] and the right child of 1[2] (at frame #3) merges into one node. This merged node's tempo has a Gaussian mixture distribution with two components from the two copies of the 2[1], and is approximated by a single Gaussian (thicker blue line). The two merges at frame #5 have similar approximations.

Under this strategy, it's guaranteed that a left child has no duplicates at a frame. All left children's ages are at least two (frames), because they represent continuing notes. Merging nodes are always the right children of their parent nodes, and these nodes always have the age of 1. For nodes identified as $k_n[1]$ at frame n, their parent nodes must have the label of $k_n - 1$ and could have the age of 1, 2, $\ldots$, $n - k_n$ from frame $n-1$. Therefore, there are $n - k_n$ copies of nodes $k_n[1]$ being merged together at frame n. It's worth noting that the size of the merging group, $n - k_n$, could be 1, which means there is no merging happening.

Distinguishing those two cases, we can iteratively approximate the discrete filtered probability of the label and the age, $p(k_n, a_n | y_1^n)$, and the continuous filtered probability of the tempo, $p(t_{k_n} | k_n, a_n, y_1^n)$. From previous discussions, we know that $p(t_{k_n} | k_n, a_n, y_1^n)$ is always a single Gaussian (after approximation), and we write its mean and variance as $\mu_n = \mu(k_n, a_n, y_1^n)$ and $\sigma_n^2 = \sigma^2(k_n, a_n, y_1^n)$.

The discrete and the continuous filtered probabilities can be iteratively calculated as follows in two cases. In the case where it stays in the same note, so $k_n = k_{n-1}$ and $a_n > 1$, there is only one parent node (without merging):

$$\begin{aligned} p(k_n, a_n | y_1^n) &= \frac{1}{W_n} p(k_n, a_n, y_n | y_1^{n-1}) \\ &= \frac{1}{W_n} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \end{aligned} \quad (14)$$

$$\begin{aligned} &g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) = \quad (15) \\ &p(k_{n-1}, a_{n-1} | y_1^{n-1}) p(k_n, a_n | k_{n-1}, a_{n-1}, y_1^{n-1}) p(y_n | k_n) \\ &W_n = p(y_n | y_1^{n-1}) = \sum_{\{k_n, a_n\}} p(k_n, a_n, y_n | y_1^{n-1}) \end{aligned}$$

The middle factor in Equation 15 has been discussed in Equations 9 and 12, and the third factor is the data model. The filtered tempo in this case doesn't change, i.e.,

$$p(t_{k_n} | k_n, a_n, y_1^n) = p(t_{k_{n-1}} | k_{n-1}, a_{n-1}, y_1^{n-1}) \quad (16)$$

In the other case where it transitions to the next note, so $k_n = k_{n-1} + 1$ and $a_n = 1$, the only difference for the discrete filtered probability is that there is a merging process represented by the summation operation:

$$p(k_n, a_n | y_1^n) = \frac{1}{W_n} \sum_{\substack{1 \le a_{n-1} \\ \le n - k_n}} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \quad (17)$$

The filtered tempo becomes a Gaussian mixture:

$$\begin{aligned} &p(t_{k_n} | k_n, a_n, y_1^n) \quad (18) \\ &= \frac{1}{p(k_n, a_n | y_1^n) W_n} \sum_{\substack{1 \le a_{n-1} \\ \le n - k_n}} g(k_{n-1}, a_{n-1}, k_n, a_n, y_1^n) \cdot \\ &\qquad p(t_{k_n} | o_{k_n} = n, o_{k_{n-1}} = n - a_{n-1}, \mu_{n-1}, \sigma_{n-1}^2) \end{aligned}$$

The last term can be calculated by the Kalman filter in Section 3.1. We further approximate the above Gaussian mixture using a single Gaussian with the mean as μ_n and the variance as σ_n^2. Putting Equations 14-18 together, we can iteratively calculate $p(k_n, a_n, t_{k_n} | y_1^n)$ by

$$p(k_n, a_n, t_{k_n} | y_1^n) = p(k_n, a_n | y_1^n)\, p(t_{k_n} | k_n, a_n, y_1^n)$$

5. PRELIMINARY EXPERIMENTS

5.1 Data Set

In score-following , the ground truth is the note onset times in the performance audio. This is usually difficult to obtain without tedious work of hand labeling or correction. One idea is to record the performances on pianos that capture movements of the keys, hammers and pedals, and store the information in MIDI files (e.g., *Disklavier* pianos). We can infer the note onsets from a MIDI file fairly easily. The MAESTRO data set by [19] contains such audio and MIDI data from 172 hours of piano performances from the International Piano-e-Competition [20].

The preliminary experiments contained 50 excerpts of real performances from 14 solo-piano pieces, as shown in Table 1. A typical excerpt lasted from 40 sec. to 90 sec., making the entire data set 48 min. of music. About 33 min. of it was from the MAESTRO, and the rest from the publicly available music recordings on the Internet. In the former case, we matched the performance MIDI data with the digital score according to the minimum-edit-distances criterion, with some minor manual corrections. For the latter case, we ran an offline audio-to-score algorithm which generated "close to perfect" results, and then manually corrected them. The audio data, along with a detailed description of the measures in these pieces are available at http://music.informatics.indiana.edu/papers/ismir20/.

Composer	Piece	#Excerpts
Mozart	Piano Concerto No. 17 in G major, mvmt1	3
Schumann	Piano Concerto in A minor, mvmt1	3
Chopin	Barcarolle, Op. 60	2
Chopin	Prelude, Op. 28 No. 4	2
Chopin	Ballade No. 1	8
Liszt	*La campanella*	5
Rachmaninoff	Prelude, Op. 3, No. 2	5
Schubert	Six Moments, D. 780 No. 2	1
Schubert	Ständchen, D 957 No. 4 from Schwanengesang	4
Debussy	Prelude, No. 2 (Voiles)	1
Debussy	*La fille aux cheveux de lin*	3
Beethoven	Piano Sonata No. 8 (Sonata Pathétique)	1
Beethoven	Piano Sonata No. 31	8
Haydn	Piano Sonata No. 24 in D major, mvmt1	1
Haydn	Piano Sonata No. 24 in D major, mvmt2&3	3

Table 1. Piano music used in the experiments.

5.2 Evaluation Method

Write $\kappa_1, \ldots, \kappa_N$ as the ground truth index labels of all audio frames. At any frame, the probability of recognizing the truth note is then the sum of the filtered probabilities with the correct label (regardless of age), as in Equation 19. The overall accuracy on an excerpt is the average across all frames. This is called the *frame-wise accuracy* [14], accounting for the accuracy of every frame.

$$Acc_n = \sum_{\substack{k_n=\kappa_n \\ 1\le a_n\le n-\kappa_n+1}} p(k_n, a_n|y_1^n) \qquad (19)$$

$$Acc = \sum\nolimits_n Acc_n/N \qquad (20)$$

5.3 Results

The baseline algorithm for comparison was Music Plus One [21], a state-of-the-art score-following systems, based on a hidden Markov model. Both algorithms used $8kHz$ sampling, 512-sample frame size (64 ms), and 128-sample hop size. Both algorithms also deployed the *beam search* technique to limit hypotheses at each frame to $\le$200.

Out of the 50 excerpts, 12 excerpts had "very low" accuracies (< 40%) by at least one of the two algorithms. Very low accuracy can either exemplify a fatal error, in which the program got lost at certain frames, and never found its way back, or it can mean high uncertainty among neighboring chords. As shown in Table 2, the proposed method failed exactly two times fewer than the baseline, and it also had higher average accuracy among the failed excerpts.

	baseline	proposed
# failed excerpts	11	9
average accuracy	15.1%	22.1%

Table 2. Counts of low-accuracy excerpts.

Excluding those 12 excerpts, we calculated the average overall accuracy across all of the other 38 excerpts, as shown in Table 3. The proposed method achieved 4.1% higher accuracy than the baseline. A p-value of .0096 ($\alpha <$ .01) on a paired t-test indicates that the proposed method is measurably better than the baseline.

	baseline	proposed
average accuracy	65.0%	69.1%

Table 3. Average accuracies of 38 excerpts.

6. DISCUSSION AND CONCLUSION

Piano music is one of the most challenging cases in the score-following realm, as the preliminary experiments indicate: the baseline algorithm failed on 22% of excerpts. Further investigation suggests that the proposed method is more robust than the baseline: fewer fatal errors, easier recovery from mistakes, and successful following even when the performance tempo was far from the default tempo. With this evidence and significantly improved accuracy on successfully followed excerpts, we can speculate that treating the tempo as a variable helps the program adapt to unpredictable performance variations, and that modeling the tempo as smooth helps discriminate among hypotheses.

Although the dataset is not large enough to draw general conclusions, the preliminary experiments showed strong promise in the direction of tracking tempo while following a score. The presented method here is general and can be applied to a variety of instruments, monophonic or polyphonic. The tracked tempo information is also meaningful for anticipating the next note in automatic accompaniment systems, and is scalable to analyze the timing aspects of large numbers of performances.

In conclusion, this paper presents an interesting new method for improved score-following, and suggests a promising direction for future research endeavors.

7. REFERENCES

[1] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening." in *ECAI*, 2008, pp. 241–245.

[2] R. B. Dannenberg and C. Raphael, "Music score alignment and computer accompaniment," *Communications of the ACM*, vol. 49, no. 8, pp. 38–43, 2006.

[3] A. Cont, "On the creative use of score following and its impact on research," in *SMC*, 2011.

[4] E. Schoonderwaldt, A. Askenfelt, and K. F. Hansen, "Design and implementation of automatic evaluation of recorder performance in IMUTUS," in *In Proceedings of the International Computer Music Conference (ICMC)*, 2005.

[5] A. Cont, "A coupled duration-focused architecture for real-time music-to-score alignment," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 974–987, 2010.

[6] P. Cuvillier, "On temporal coherency of probabilistic models for audio-to-score alignment," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2016.

[7] C. Joder, S. Essid, and G. Richard, "A conditional random field viewpoint of symbolic audio-to-score matching," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 871–874.

[8] ——, "Hidden discrete tempo model: A tempo-aware timing model for audio-to-score alignment," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 397–400.

[9] N. Montecchio and A. Cont, "A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques," in *ICASSP 2011: Proceedings of International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 193–196.

[10] T. Otsuka, K. Nakadai, T. Takahashi, T. Ogata, and H. Okuno, "Real-time audio-to-score alignment using particle filter for coplayer music robots," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, p. 384651, 2011.

[11] F. Korzeniowski, F. Krebs, A. Arzt, and G. Widmer, "Tracking rests and tempo changes: Improved score following with particle filters," in *ICMC*, 2013.

[12] Z. Duan and B. Pardo, "A state space model for online polyphonic audio-score alignment," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 197–200.

[13] C. Raphael, "Aligning music audio with symbolic scores using a hybrid graphical model," *Machine learning*, vol. 65, no. 2-3, pp. 389–409, 2006.

[14] Y. Jiang and C. Raphael, "Piano score-following by tracking note evolution." in *SMC*, 2019.

[15] K. Gröchenig, *Foundations of time-frequency analysis*. Springer Science & Business Media, 2013.

[16] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. Oxford university press, 2012.

[17] Z. Ghahramani and G. E. Hinton, "Variational learning for switching state-space models," *Neural computation*, vol. 12, no. 4, pp. 831–864, 2000.

[18] A. R. Runnalls, "Kullback-Leibler approach to Gaussian mixture reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, 2007.

[19] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," 2018.

[20] "International piano-e-competition," http://piano-e-competition.com.

[21] C. Raphael, "Music Plus One and Machine Learning." in *ICML*, 2010, pp. 21–28.