A Hybrid Graphical Model for Rhythmic Parsing

Christopher Raphael*
Department of Mathematics and Statistics
University of Massachusetts, Amherst
raphael@math.umass.edu

October 3, 2001

Abstract

A method is presented for the rhythmic parsing problem: Given a sequence of observed musical note onset times, we simultaneously estimate the corresponding notated rhythm and tempo process. A graphical model is developed that represents the evolution of tempo and rhythm and relates these hidden quantities to an observable performance. The rhythm variables are discrete and the tempo and observation variables are continuous. We show how to compute the globally most likely configuration of the tempo and rhythm variables given an observation of note onset times. Experiments are presented on both MIDI data and a data set derived from an audio signal. A generalization to computing MAP estimates for arbitrary conditional Gaussian distributions is outlined.

1 Introduction

Rhythm is the aspect of music that deals with *when* events occur. Typically, rhythm in Western music is notated in a way that expresses the position of each note as a rational number, usually in terms of some relatively small common denominator. For instance, if we use the *measure* as our unit of notated position, then the sequence of measure positions $m_0 = 0, m_1 = 1/4, m_2 = 1, \ldots$ expresses the notion that the first note occurs at the beginning of the 1st measure, the second note occurs 1/4 the way through the 1st measure, the third note occurs at the beginning of the 2nd measure, etc. If the music is performed with mechanical precision then a single number, the *tempo*, will map the measure positions to actual times. For instance, if the tempo is 3 seconds per measure, then the notes would occur at 0 secs, 3/4 secs, 3 secs, etc. However, such a performance would be nearly impossible for the human perform to create, and, moreover, would be undesirable. Much of the expressively quality of a musical performance comes from the way in which the actual note times deviate from what is prescribed by a literal interpretation of the printed music. In particular, there are two primary components to this *expressive timing* [1]. Firstly, the actual tempo is often not constant, but rather continually varied throughout the evolution of the performance. Secondly, there are more local (note by note) distortions which can be accidental, or can result from interpretive considerations.

We focus here on a problem encountered in music information retrieval (MIR): Given a sequence of measured note onset times, we wish to identify the corresponding sequence of measure positions. We call this process *rhythmic parsing*. The time sequences forming the input to our procedure could be estimated from an audio signal or could come directly from a MIDI (musical instrument digital interface) file — a sequence of time-tagged musical events such as note beginnings and endings. For example, consider the data in the left panel of Figure 1 containing estimated note times from an excerpt of Schumann's 2nd Romance for oboe and piano (oboe part only). The actual audio file can be heard at http://fafner.math.umass.edu/rhythmic_parsing. Our goal is to assign the proper score

*This work is supported by NSF grants IIS-0113496 and IIS-9987898.

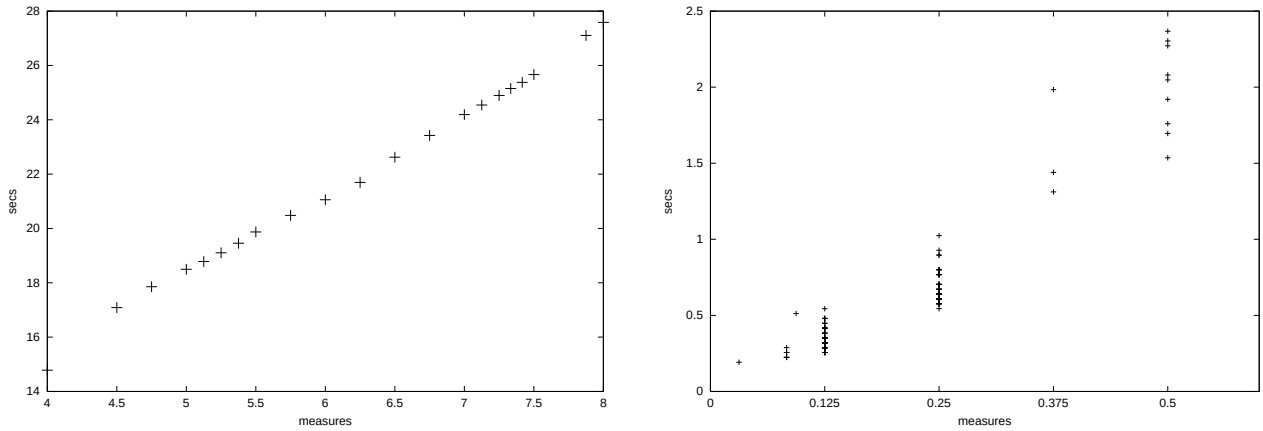

Figure 1: **Left:** Real time (seconds) vs. Musical time (measures) for the Schumann data. **Right:** The actual durations (seconds) of notes grouped by the musical duration (measures).

position, in measures, to each the observed times. When this is done correctly, as in Figure 1, the observed times, in seconds, plotted against the score positions, in measures, trace out a curve whose local slope gives the player's local tempo.

Applications of rhythmic parsing are numerous. Virtually every commercial score-writing program now offers the option of creating scores by directly entering MIDI data from a keyboard. Such programs must infer the rhythmic content from the actual times at which musical events occur and, hence, must address the rhythmic parsing problem. When the input data is played with anything less than mechanical precision, the transcription degrades rapidly, due to the difficulty in computing the correct rhythmic parse. Rhythmic parsing also has applications in musicology where it could be used to separate the inherently intertwined quantities of notated rhythm and expressive timing. Either the rhythmic data or the timing information could be the focal point of further study. Additionally, several applications of rhythmic parsing are related to efforts in music information retrieval, as follows. The musical world eagerly awaits the compilation of music databases containing virtually every kind of (public domain) music, thereby facilitating the searching, studying, comparing, and understanding of music. The construction of such data bases will likely involve several transcription efforts including optical music recognition, musical audio signal recognition, and MIDI transcription. Rhythmic parsing is an essential ingredient to the latter two efforts. Finally, the last decade has seen a virtual explosion in music data available on the World Wide Web. Unfortunately, content-based searches analogous to those performed on text are not possible at the present time. However, if automated music transcription were to progress to a sufficient level, searchable descriptions of musical content could be constructed automatically. Such a development would dramatically increase access to music on the web. Rhythmic parsing will play a significant role in this endeavor too.

As already mentioned, mostly commercial score-writing address the rhythmic parsing problem. Usually these efforts attempt to *quantize* the observed note lengths, or more precisely inter-onset intervals (IOIs), to their closest note values (eighth note, quarter note, etc.), given a known tempo, or to quantize the observed note onset times to the closest points in a rigid grid [2]. While such quantization schemes can work reasonably well when the music is played with robotic precision (often a metronome is used), they perform poorly when faced with the more expressive and less accurate playing typically encountered. Consider the right panel of Figure 1 in which we have plotted the written note lengths in measures versus the actual note lengths (IOIs) in seconds from our musical excerpt. The large degree of overlap between the empirical distributions of each note length class demonstrates the futility of assigning note lengths through note-by-note quantization in this example. In this particular example, the overlap in empirical distributions is mostly attributable to tempo fluctuations in the performance.

In addition to the commercial systems, we are also aware of several research efforts related to rhythm transcription. Some of this research addresses the problem of *beat induction*, or *tempo*

tracking in which one tries to accomplish the equivalent of "foot-tapping" — estimating a sequence of times corresponding to evenly spaced rhythmic intervals (e.g. beats) for a given sequence of observed note onset times [3], [4], [5], [6], [7], [8], [9] [10]. A complementary research effort addresses the problem of assigning rhythmic values as simple integer ratios to observed note lengths without any corresponding estimation of tempo [1], [11], [12]. The latter two assume that beat induction has already been performed, where as the former assumes that tempo variations are not significant enough to obscure the ratios of neighboring note lengths.

In many kinds of music we believe it will be exceedingly difficult to *independently* estimate tempo and rhythm, as in the previously cited research, since the observed data is formed from a complex interplay between the two. That is, independent estimation of tempo or rhythm leads to a "chicken and egg" problem: One cannot easily estimate rhythm without knowing tempo and vice-versa. In this work we address the problem of *simultaneous* estimation of tempo and rhythm. From a problem domain point of view, this is the most significant contrast between our work and other efforts cited.

The research effort closest to ours in spirit is the recent work of Cemgil [13] which probabilistically models tempo and rhythm jointly and seeks globally optimal data interpretations by computing the posterior distribution through particle filtering techniques. There are two significant distinctions between this work and ours. Cemgil deals with the "chicken and egg" problem by approximating the marginal distribution on rhythm by integrating out the tempo variables; we instead estimate tempo and rhythm jointly. Perhaps a more important distinction is that we provide a dynamic programming technique that identifies the globally optimal data interpretation; Cemgil's method is approximate.

The paper is organized as follows. Section 2 develops a generative graphical model for the simultaneous evolution of tempo and rhythm processes that incorporates both prior knowledge concerning the nature of the rhythm process and a simple and reasonable model for tempo evolution. This section then describes a computational scheme for identifying the most likely configuration, a MAP estimate, of the unobserved processes given observed musical data. Section 3 demonstrates the application of our scheme to a several musical examples. Section 4 then briefly summarizes and discusses some aspects of our approach to rhythmic parsing. Our method of identifying the MAP estimate for the specific model treated here generalizes well beyond this particular model, however. Section 5 sketches the generalization of our methodology to the generic MAP estimation of unobserved variables for conditional Gaussian (CG) distributions. To our knowledge, MAP estimation in CG distributions has not be studied previously, however is potentially quite useful. Finally, the appendix lists some easily-derived results about Gaussian kernels that are used in preceding sections.

2 Rhythmic Parsing

2.1 The Model

While musical rhythm is not usually composed of rhythmic fragments that repeat verbatim, rhythm typically has a cyclic component in which certain tendencies repeat in a periodic fashion. This periodic nature of music is so basic that it figures prominently in the way most Western music is notated: As a sequence of *measures* — units of musical time — that obey similar subdivision rules. The probabilistic modeling of this periodic behavior is central to the approach taken here, and we will present evidence in Section 3 of its advantage. In what follows we use the term *measure* to denote the most obvious period of the rhythmic structure. Usually this will be the same as the notated measure.

Suppose a musical instrument generates a sequence of times $o_0, o_1, \ldots, o_N$, in seconds, at which note onsets occur. Suppose we also have a finite set $\mathcal{S}$ composed of the possible *measure positions* a note can occupy. For instance, if the music is in $4/4$ time and we believe that no subdivision occurs beyond the eighth note, then $\mathcal{S} = \{i/8 : i = 0, \ldots, 7\}$. More complicated subdivision rules could lead to sets, $\mathcal{S}$, which are not evenly spaced multiples of some common denominator. We assume only that the possible onset positions of $\mathcal{S}$ are rational numbers in $[0, 1)$, decided upon in advance. Our goal is to associate each note onset o_n with a score position — a measure number and an element of $\mathcal{S}$.

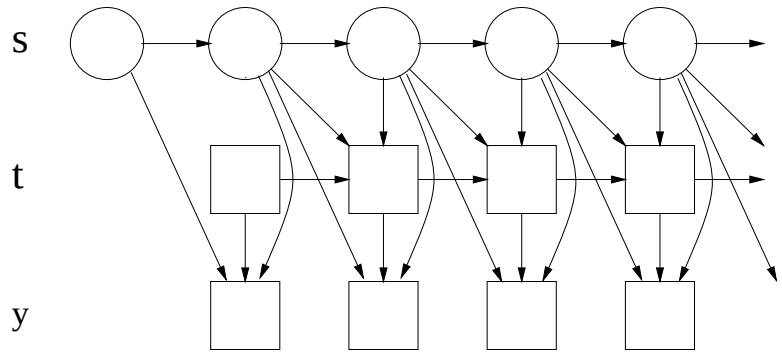

Figure 2: The DAG describing the dependency structure of the variables of our model. Circles represent discrete variables while squares represent continuous variables.

We model this situation as follows. Let $S_0, S_1, \ldots, S_N$ be the discrete measure position process, $S_n \in \mathcal{S}, n = 0, \ldots, N$. In interpreting these positions we assume that each consecutive pair of positions differs by less than one measure. For instance, in the 4/4 example given above $S_n = 0/8, S_{n+1} = 1/8$ would mean the nth note begins at the start of the measure and the $(n+1)$th note begins one eighth note later, while $S_n = 0/8, S_{n+1} = 0/8$ would mean that the two notes both begin at the start of the *same* measure. We can then use

$$l(s_n, s_{n+1}) = \begin{cases} s_{n+1} - s_n & \text{if } s_{n+1} \geq s_n \\ 1 + s_{n+1} - s_n & \text{otherwise} \end{cases}$$

to unambiguously represent the gap, in measures, associated with the transition from s_n to s_{n+1}. Thus, if $s_0, s_1, \ldots, s_N$ is known, we assign a score position, m_n, to every observation o_n by $m_n = s_0 + \sum_{\nu=0}^{n-1} l(s_\nu, s_{\nu+1})$. We believe the assumption that inter-onset intervals are less than one measure is appropriate for many examples — especially those in which the *composite* rhythm generated by superposing the musical parts is studied, as in the Chopin example of Section 3. However, more complicated models can allow for longer IOIs without greatly increasing the number of parameters to be learned. We model the S process as a time-homogeneous Markov chain with initial distribution

$$I(s_0) = P(S_0 = s_0)$$

and transition probability matrix

$$R(s_n, s_{n+1}) = P(S_{n+1} = s_{n+1} | S_n = s_n)$$

The tempo is the most important link between what is prescribed by the score and what is observed. Let $T_1, T_2, \ldots, T_N$ be the continuously-valued tempo process, measured in *seconds per measure*, which we model by

$$T_1 \sim N(\nu, \phi^2)$$

and

$$T_n = T_{n-1} + \delta_n$$

for $n = 2, 3, \ldots, N$ where $\delta_n \sim N(0, \tau^2(S_{n-1}, S_n))$. This model captures the property that the tempo tends to vary smoothly and allows the variance in the tempo increment to depend on the transition from S_{n-1} to S_n. For instance, we would expect greater variability to be associated with longer transitions.

Finally we assume that the observed inter-onset intervals (IOI) $y_n = o_n - o_{n-1}$ for $n = 1, 2, \ldots, N$ are approximated by the product of $l(S_{n-1}, S_n)$ (measures) and T_n (secs. per measure). Specifically

$$Y_n = l(S_{n-1}, S_n) T_n + \epsilon_n$$

where $\epsilon_n \sim N(0, \rho^2(S_{n-1}, S_n))$. Note that the observation variance is also allowed to depend on the transition which can capture the notion that long transitions will be associated with greater variability of the IOIs. The variables $\{T_1, \delta_2, \ldots, \delta_N, \epsilon_1, \ldots, \epsilon_N\}$ are assumed to be mutually independent with T_1 independent of S_0, and δ_n and ϵ_n independent of $S_0, \ldots, S_{n-1}$.

These modeling assumptions lead to a graphical model whose directed acyclic graph is given in Figure 2. The model is composed of both discrete and Gaussian variables with the property that, for every configuration of discrete variables, the continuous variables have a multivariate Gaussian distribution. Thus, the $S_0, \ldots, S_N$, $T_1, \ldots, T_N$, $Y_1, \ldots, Y_N$ collectively have a conditional Gaussian (CG) distribution.

Such distributions were introduced by Lauritzen and Wermuth [14], [15], and have been developed by Lauritzen [16], and Lauritzen and Jensen [17], in which evidence propagation methodology is described, enabling the computation of local marginal distributions. Using these ideas, we could, in principle, fix $Y_1 = y_1, \ldots, Y_N = y_N$ and proceed to compute marginal distributions on the $\{S_n\}$ and choose as our estimate of S_n

$$\bar{s}_n = \arg\max_{s \in \mathcal{S}} P(S_n = s | Y_1 = y_1, \ldots, Y_N = y_N)$$

However, these computations rely on construction of a triangulated graph with a *strong root*. The additional edges involved in the construction of such a strong root leads to a graph in which a single clique contains the entire collection of $\{S_n\}$ variables [18]. The following computations are intractable. Furthermore, there is no guarantee that the sequence $\bar{s}_0, \ldots, \bar{s}_N$ is reasonable, or even that

$$P(S_0 = \bar{s}_0, \ldots, S_N = \bar{s}_N | Y_1 = y_1, \ldots, Y_N = y_N) > 0$$

calling this estimate into question.

Rather, we desire the *configuration* of unobserved variables which has greatest probability given the observation. Thus, regarding $y_1, \ldots, y_N$ as fixed, we seek the estimate

$$(\hat{s}, \hat{t}) = \arg\max_{s,t} L(s, t, y) \tag{1}$$

where $L(s, t, y)$ is the joint likelihood of $s = (s_0, \ldots, s_N)$, $t = (t_1, \ldots, t_N)$, $y = (y_1, \ldots, y_N)$.

The computation of such MAP estimators for networks composed entirely of discrete variables is well-known [19],[20]. In what follows we demonstrate new methodology for the exact computation of the global maximizer, $(\hat{s}, \hat{t})$ in our mixed discrete and continuous case.

2.2 Computing the Rhythmic Parse

Define the n-dimensional *Gaussian kernel*

$$K(x; \theta) = K(x; h, m, Q) = h e^{-\frac{1}{2}(x-m)^t Q (x-m)} \tag{2}$$

where x is an n-vector, h is a nonnegative constant, m is an n-vector, and Q is an $n \times n$ nonnegative definite matrix. Note that we do *not* require Q to be invertible, hence the function $K(x; \theta)$ does not necessarily correspond to a scaled Gaussian density function. We write $\theta = (h(\theta), m(\theta), Q(\theta))$ to represent the components of θ. It is possible to perform a number of operations on Gaussian kernels such as multiplication of two kernels, maximizing over a subset of variables, and representing conditional Gaussian distributions by performing transformations of the parameters involved. The appendix gives an account of some easily derived results involving Gaussian kernels.

The joint likelihood function $L(s, t, y)$ with y held fixed can be represented as follows. We define

$$\begin{aligned} L_1(s_0, s_1, t_1) &= I(s_0) R(s_0, s_1) N(t_1; \nu, \phi^2) N(y_1; l(s_0, s_1) t_1, \rho^2(s_0, s_1)) \\ &= K(t_1; \theta'(s_0, s_1)) \end{aligned} \tag{3}$$

where $N(\cdot\,; \mu, \sigma^2) = K(\cdot\,; (2\pi\sigma^2)^{-1/2}, \mu, 1/\sigma^2)$ is the univariate normal density function. In Eqn. 3, $\theta'(s_0, s_1)$ is computed for each configuration of s_0, s_1 by representing the conditional density for y_1

as a Gaussian kernel in y_1 and t_1 using Eqn. 26, eliminating y_1 from the same kernel by holding it fixed using Eqn. 24, multiplying the two kernels together using Eqn. 20, and absorbing the two constants $I(s_0)$ and $R(s_0, s_1)$ into $h(\theta'(s_0, s_1))$. Using the notation $a_i^j = (a_i, a_{i+1}, \ldots, a_j)$, we then define

$$L_n(s_0^n, t_1^n) = L_{n-1}(s_0^{n-1}, t_1^{n-1}) C_n(s_{n-1}, s_n, t_{n-1}, t_n)$$

for $n = 2, \ldots, N$ where

$$\begin{aligned} C_n(s_{n-1}, s_n, t_{n-1}, t_n) &= R(s_{n-1}, s_n) N(t_n; t_{n-1}, \tau^2(s_{n-1}, s_n)) N(y_n; l(s_{n-1}, s_n) t_n, \rho^2(s_{n-1}, s_n)) \\ &= K(t_{n-1}, t_n; \theta_n^c(s_{n-1}, s_n)) \end{aligned} \tag{4}$$

where Eqn. 4 is computed by representing the two conditional normal densities as Gaussian kernels using Eqn. 26, eliminating y_n from the second density using Eqn. 24, extending the second density to be a function of t_n and t_{n-1} using Eqn. 25, and multiplying the two factors together using Eqn. 20, and absorbing the constant $R(s_{n-1}, s_n)$.

Note that $L_N(s_0^N, t_1^N)$ is the joint likelihood $L(s, t, y)$ with y held fixed to the vector of observations. We will compute our MAP estimate by maximizing L_N using dynamic programming as follows.

Define

$$\begin{aligned} H_1(s_1, t_1) &= \max_{s_0} L_1(s_0, s_1, t_1) \\ H_n(s_n, t_n) &= \max_{s_0^{n-1}, t_1^{n-1}} L_n(s_0^n, t_1^n) \\ &= \max_{s_0^{n-1}, t_1^{n-1}} L_n(s_0^{n-1}, t_1^{n-1}, s_n, t_n) \end{aligned}$$

for $n = 2, \ldots, N$. The fundamental observation of dynamic programming is that we can compute H_n recursively by

$$\begin{aligned} H_{n+1}(s_{n+1}, t_{n+1}) &= \max_{s_0^n, t_1^n} L_{n+1}(s_0^{n+1}, t_1^{n+1}) \\ &= \max_{s_0^n, t_1^n} L_n(s_0^n, t_1^n) C_n(s_n, s_{n+1}, t_n, t_{n+1}) \\ &= \max_{s_n, t_n} H_n(s_n, t_n) C_n(s_n, s_{n+1}, t_n, t_{n+1}) \end{aligned} \tag{5}$$

for $n = 1, \ldots, N-1$.

Consider first the computation of $H_1(s_1, t_1)$ which can be computed by "maxing out" the s_0 variable in Eqn. 3. Thus

$$\begin{aligned} H_1(s_1, t_1) &= \max_{s_0} L_1(s_0, s_1, t_1) \\ &= \max_{s_0} K(t_1; \theta'(s_0, s_1)) \\ &= \max_{\theta_1 \in \tilde{\Theta}_1(s_1)} K(t_1; \theta_1) \qquad (6) \\ &= \max_{\theta_1 \in \Theta_1(s_1)} K(t_1; \theta_1) \qquad (7) \end{aligned}$$

where

$$\tilde{\Theta}_1(s_1) = \{\theta'(s_0, s_1) : s_0 \in \mathcal{S}\}$$

and $\Theta_1(s_1) = \mathrm{Thin}(\tilde{\Theta}(s_1))$ where $\mathrm{Thin}(\Theta)$ is the smallest subset of Θ such that

$$\max_{\theta \in \mathrm{Thin}(\Theta)} K(t; \theta) = \max_{\theta \in \Theta} K(t; \theta) \tag{8}$$

This computation is depicted in Figure 3.

We remark that it is a simple matter to identify $\Theta_1(s_1) = \mathrm{Thin}(\tilde{\Theta}(s_1))$ since we can "build" the maximum of Eqn. 6 by incrementally adding components of $\tilde{\Theta}_1(s_1)$ while discarding those that leave the maximum unchanged. This algorithm is made more precise in Section 2.3.

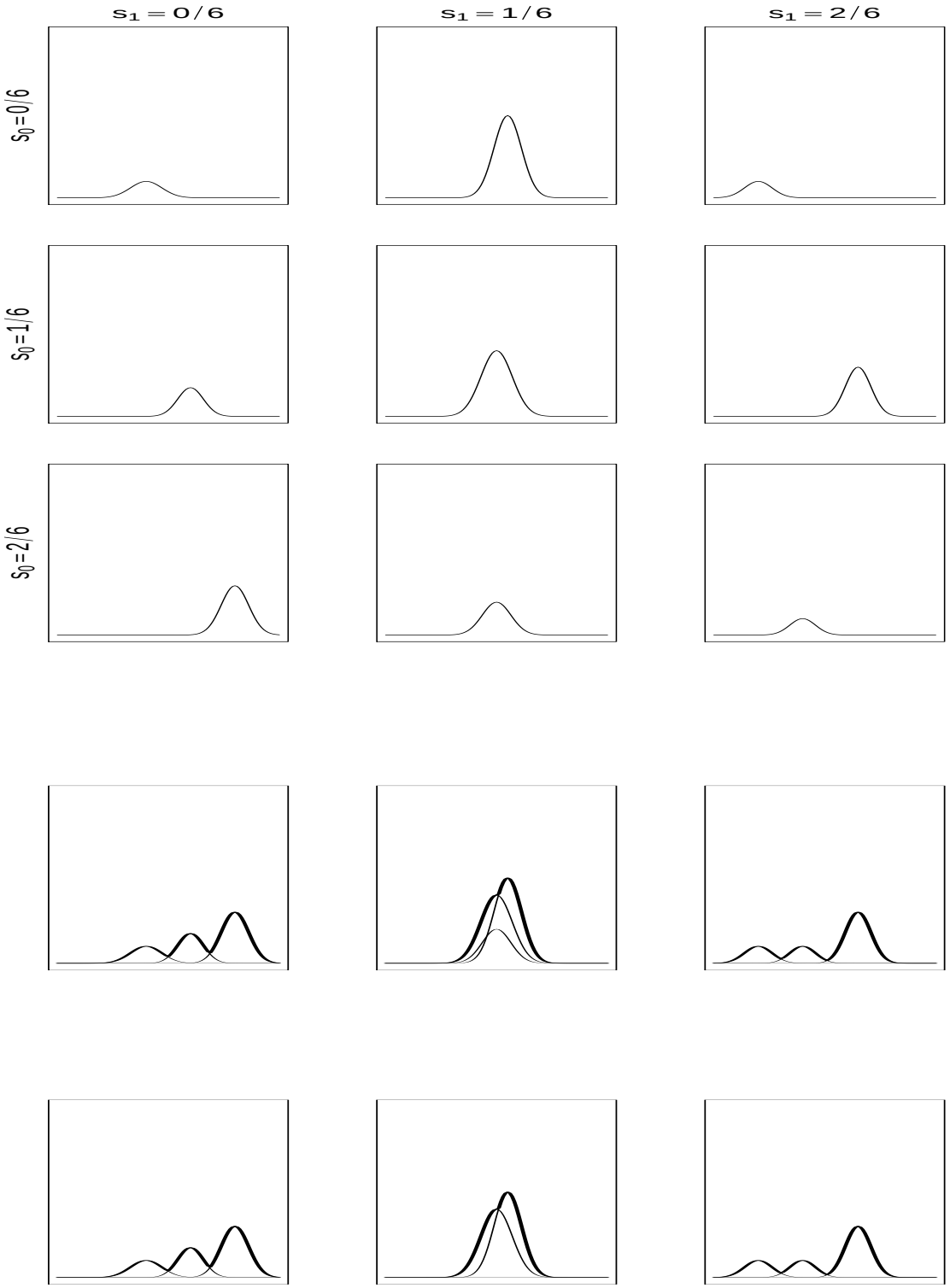

Figure 3: The construction of $H_1(s_1, t_1)$. **Top (9-panel):** The graph in the s_0, s_1 position gives $L(s_0, s_1, t_1)$. **Middle (3-panel):** Maxing out over s_0 corresponds to superimposing the graphs in a column and taking the maximum (shown in bold). **Bottom (3-panel):** The thinning operation removes kernels from the representation without affecting the maximum. Note that the middle plot in the bottom panel now is composed of only two kernels where before it had three.

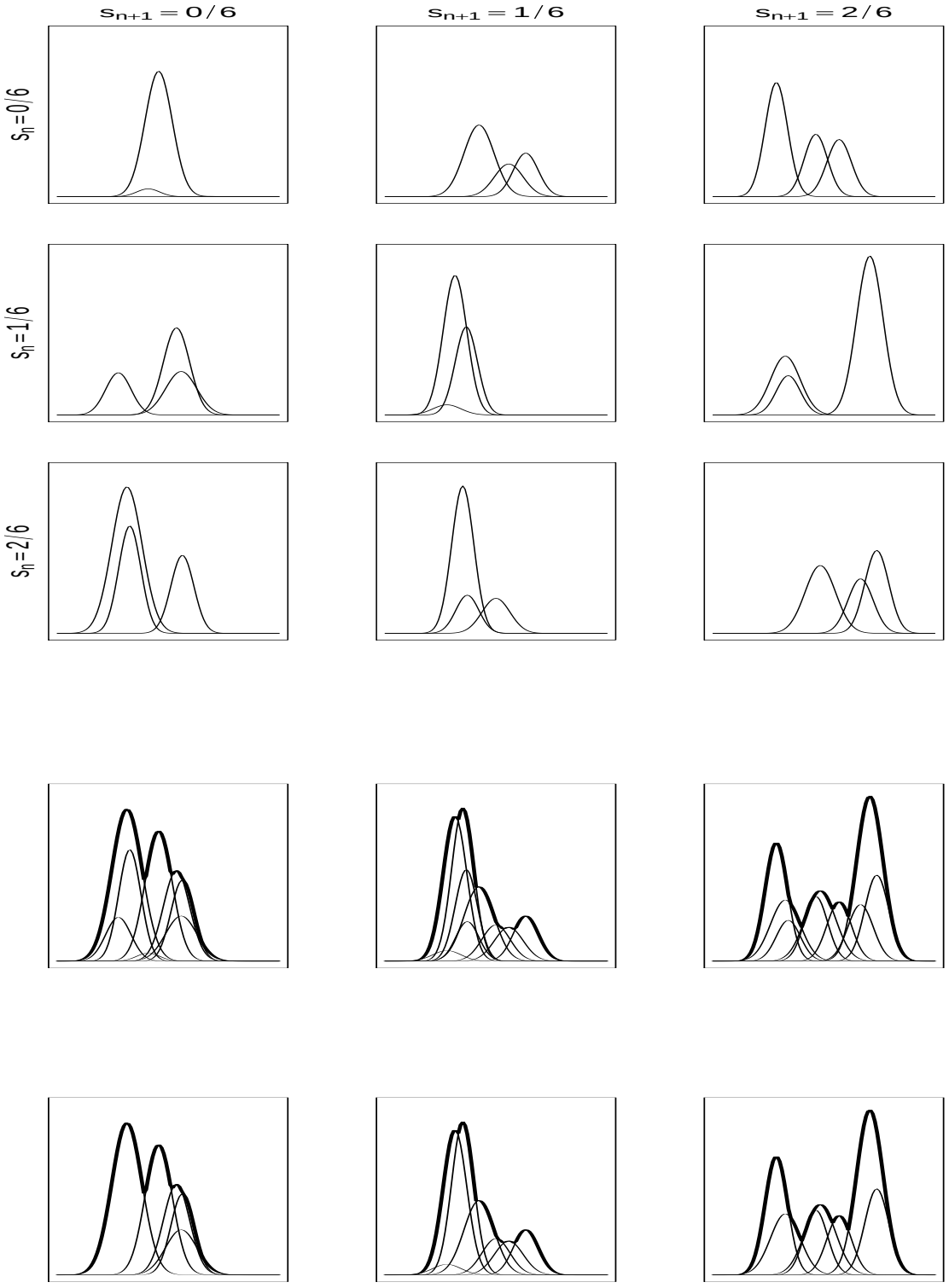

Figure 4: The construction of $H_{n+1}(s_{n+1}, t_{n+1})$. **Top (9-panel):** $\max_{t_n} H_n(s_n, t_n) C_{n+1}(s_n, s_{n+1})$ is depicted. The continuous variable is t_{n+1}. **Middle (3-panel):** Maxing out over s_n gives $H_{n+1}(s_{n+1}, t_{n+1})(shown in bold)$ **Bottom (3-panel):** $H_{n+1}(s_{n+1}, t_{n+1})$ is represented with fewer kernels after thinning.

The computational feasibility of our dynamic programming algorithm follows because the form of Eqn. 7 — a maximum of Gaussian kernels — is invariant under the operation of Eqn. 5. That is, assuming

$$H_n(s_n, t_n) = \max_{\theta_n \in \Theta_n(s_n)} K(t_n; \theta_n) \tag{9}$$

we have

$$\begin{aligned}
H_{n+1}(s_{n+1}, t_{n+1}) &= \max_{s_n, t_n} H_n(s_n, t_n) C_{n+1}(s_n, s_{n+1}, t_n, t_{n+1}) \\
&= \max_{s_n, t_n} \max_{\theta_n \in \Theta_n(s_n)} K(t_n; \theta_n) K(t_n, t_{n+1}; \theta^c_{n+1}(s_n, s_{n+1})) \\
&= \max_{s_n, \theta_n \in \Theta_n(s_n)} \max_{t_n} K(t_n; \theta_n) K(t_n, t_{n+1}; \theta^c_{n+1}(s_n, s_{n+1})) \qquad (10) \\
&= \max_{s_n, \theta_n \in \Theta_n(s_n)} K(t_{n+1}; \tilde{\theta}(\theta_n, \theta^c_{n+1}(s_n, s_{n+1}))) \qquad (11) \\
&= \max_{\theta_{n+1} \in \tilde{\Theta}_{n+1}(s_{n+1})} K(t_{n+1}; \theta_{n+1}) \\
&= \max_{\theta_{n+1} \in \Theta_{n+1}(s_{n+1})} K(t_{n+1}; \theta_{n+1})
\end{aligned}$$

where in going from Eqn. 10 to 11, i.e. in computing $\tilde{\theta}(\theta_n, \theta^c_n(s_n, s_{n+1}))$, we use Eqns. 20, 25, and 23. $\tilde{\Theta}_{n+1}(s_{n+1})$ in the preceding is given by

$$\tilde{\Theta}_{n+1}(s_{n+1}) = \{\tilde{\theta}(\theta_n, \theta^c_n(s_n, s_{n+1})) : \theta_n \in \Theta_n(s_n), s_n \in \mathcal{S}\}$$

and $\Theta_{n+1}(s_{n+1}) = \text{Thin}(\tilde{\Theta}_{n+1}(s_{n+1}))$. This computation is depicted in Figure 4

While the comparison of Eqns. 9 and 11 suggest $|\Theta_n(s_n)|$ increases exponentially with n, this growth will be controlled by the "thinning" operation. In fact, the behavior observed in our experiments, which we anticipate is typical, was that $|\Theta_n(s_n)|$ increased to a manageable number within a few dynamic programming iterations and fluctuated around that number in the following iterations. Details are given in Section 3.

2.2.1 Recovering the Optimal Parse

The maximal value of $L = L_N$ is easily computed as follows. Define

$$\Theta_n(\mathcal{S}) = \bigcup_{s_n \in S} \Theta_n(s_n) \tag{12}$$

for $n = 1, \ldots, N$ and let

$$(\hat{t}_N, \hat{\theta}_N) = \arg \max_{t_N, \theta \in \Theta_N(\mathcal{S})} K(t_N; \theta)$$

which can be computed by letting

$$\begin{aligned}
\hat{\theta}_N &= \arg \max_{\theta \in \Theta_N(\mathcal{S})} (\max_{t_N} K(t_N; \theta)) \\
&= \arg \max_{\theta \in \Theta_N(\mathcal{S})} h(\theta)
\end{aligned}$$

and taking $\hat{t}_N = m(\hat{\theta}_N)$. Then

$$\begin{aligned}
K(\hat{t}_N; \hat{\theta}_N) &= \max_{t_N} \max_{\theta \in \Theta_N(\mathcal{S})} K(t_N; \theta) \\
&= \max_{s_N, t_N} \max_{\theta \in \Theta_N(s_N)} K(t_N; \theta) \\
&= \max_{s_N, t_N} H_N(s_N, t_N) \\
&= \max_{s_0^N, t_1^N} L_N(s_0^N, t_1^N)
\end{aligned}$$

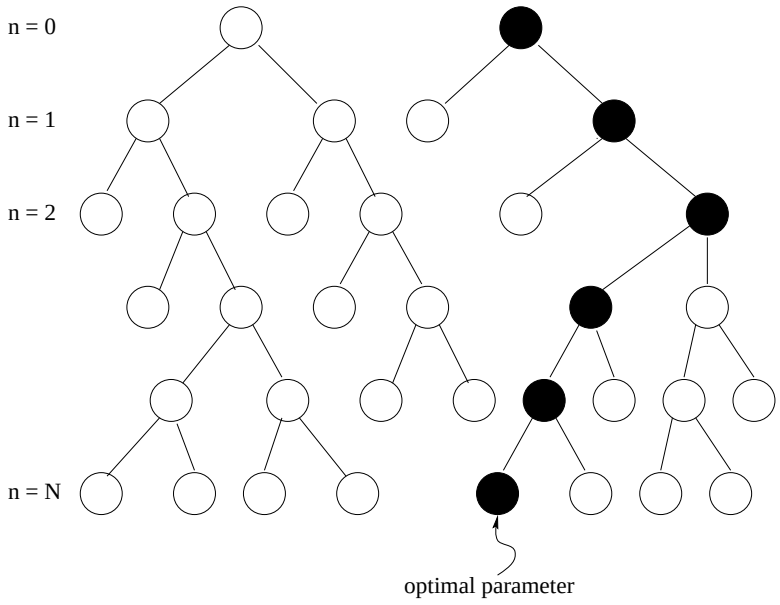

Figure 5: The figure corresponds to a situation in which $|\mathcal{S}| = 2$, thus there are two parameter values at level 0 and each parameter value has 2 child parameters where parameter values are depicted by nodes in the tree. Each parameter $\theta \in \Theta_n(\mathcal{S})$ has a unique parent, $\mathrm{Pa}(\theta)$, so the optimal sequence of parameter values $\theta_0, \ldots, \theta_N$ (shown with solid circles) can be traced back from leaf to root. $\hat{\theta}_N$ is marked as "optimal parameter" in the figure. Terminal nodes in the tree at levels other than N correspond to parameter values that have been pruned through the thinning algorithm.

Thus $K(\hat{t}_N; \hat{\theta}_N)$ is the maximal value of the likelihood function, L.

We wish to recover the rhythmic parse $\hat{s}_0^N, \hat{t}_1^N$ that attains this maximum. Considering Eqn. 11, we see that each element $\theta_{n+1} \in \Theta_{n+1}(s_{n+1})$ is generated by a unique "predecessor" or "parent" $\mathrm{Pa}(\theta_{n+1}) \in \Theta_n(\mathcal{S})$. That is, if $\theta_n \in \Theta_n(\mathcal{S})$, and

$$\theta_{n+1} = \tilde{\theta}(\theta_n, \theta_{n+1}^c(s_n, s_{n+1})) \in \Theta_{n+1}(s_{n+1})$$

then $\mathrm{Pa}(\theta_{n+1}) = \theta_n$. Thus we can trace back the optimizing sequence of parameter values by $\hat{\theta}_n = \mathrm{Pa}(\hat{\theta}_{n+1})$ for $n = 0, \ldots, N-1$ as in Figure 5. Then the optimizing sequence of measure positions in $\mathcal{S}$ is given by $\hat{s}_n = s(\hat{\theta}_n)$ for $n = 0, \ldots, N$, where $s(\theta_n) = s_n$ if $\theta_n \in \Theta_n(s_n)$.

Having identified $\hat{t}_N$ and $\hat{s}_0, \ldots, \hat{s}_N$, we can recover the optimal $\hat{t}_1, \ldots, \hat{t}_N$ through

$$\begin{aligned}
\hat{t}_n &= \arg\max_{t_n} H_n(\hat{s}_n, t_n) C_n(\hat{s}_n, \hat{s}_{n+1}, t_n, \hat{t}_{n+1}) \\
&= \arg\max_{t_n} K(t_n; \hat{\theta}_n) K(t_n, \hat{t}_{n+1}; \theta_{n+1}^c(\hat{s}_n, \hat{s}_{n+1})) \\
&= \arg\max_{t_n} K(t_n; \bar{\theta}_n) \\
&= m(\bar{\theta}_n)
\end{aligned}$$

where $\bar{\theta}_n$ can be computed by eliminating $\hat{t}_{n+1}$ using Eqn. 24 and multiplying the two kernels together using Eqn. 20.

2.3 Thinning

The computational feasibility of our dynamic programming algorithm relies on the thinning operation of Section 2.2 since without this operation the complexity of the representation of H_n grows exponentially. Recall, $\mathrm{Thin}(\Theta)$ is the smallest subset of Θ for which Eqn. 8 holds. When Θ is

composed of parameters for *one-dimensional* Gaussian kernels, as in Section 2.2, the algorithm for computing $\mathrm{Thin}(\Theta)$ is straightforward, as follows.

Suppose $\Theta = \{\theta^1, \ldots, \theta^I\}$. Define

$$\hat{\theta}^i(t) \quad = \quad \arg \max_{\theta \in \{\theta^1, \ldots, \theta^i\}} K(t; \theta)$$

for $i = 1 \ldots I$ and note that $\hat{\theta}^i(t)$ is piecewise constant and, hence, can be written as

$$\hat{\theta}^i(t) = \sum_{k=1}^{N(i)} \theta_k^i 1_{(x_k^i, x_{k+1}^i)}(t)$$

where $-\infty = x_1^i < x_2^i < \ldots < x_{N(i)}^i < x_{N(i)+1} = \infty$ and $\theta_k^i \in \{\theta^1, \ldots, \theta^i\}$. We need not be concerned with the definition of $\hat{\theta}^i(t)$ at points, $t = x_k^i$, where the maximizer is not unique. Clearly then $\mathrm{Thin}(\Theta) = \cup_{k=1}^{N(I)} \theta_k^I$.

Note that $\hat{\theta}^i(t)$ can be computed iteratively by letting $\hat{\theta}^1(t) = \theta^1$ and noting

$$\hat{\theta}^i(t) = \arg \max_{\theta \in \{\hat{\theta}^{i-1}(t), \theta^i\}} K(t; \theta) \tag{13}$$

$\hat{\theta}^i(t)$ can then be computed on each interval $(x_k^{i-1}, x_{k+1}^{i-1})$ where it must be that $\hat{\theta}^{i-1}(t) = \theta_k^{i-1}$. To do this we simply find all solutions, t, to the quadratic equation

$$\log K(t; \theta^i) - \log K(t; \theta_k^{i-1}) = 0 \tag{14}$$

which lie in $(x_k^{i-1}, x_{k+1}^{i-1})$. These points partition the interval $(x_k^{i-1}, x_{k+1}^{i-1})$ into subintervals where $\hat{\theta}^i(t)$ must be constant so we need only identify $\hat{\theta}^i(t)$ through Eqn. 13 at any interior point of these subintervals. Having done this for each interval $(x_k^{i-1}, x_{k+1}^{i-1})$ we may find that $\hat{\theta}^i(t)$ is constant over neighboring subintervals. In such a case, the neighbors are simply merged together to form a more compact representation of $\hat{\theta}^i(t)$.

2.3.1 Constrained Optimal Parse

With a minor variation on the thinning algorithm we can, in many cases, compute a *constrained* optimal parse defined by Eqn. 1 subject to $t_{\mathrm{low}} < t_n < t_{\mathrm{high}}$ for $n = 1, \ldots, N$ and fixed constants t_{low} and t_{high}. This is a helpful restriction in our rhythmic parsing problem since we know that the tempo must always be positive and can reasonably be restricted to be less than some maximum value as well. Such a constraint will also increase the efficiency of our algorithm since it will decrease the number of kernels needed to represent H_n in Eqn. 9.

We proceed as follows. Define the *modified* thinning procedure, $\mathrm{Thin}^{\mathrm{m}}(\Theta)$, to be the minimal subset of Θ such that

$$\max_{\theta \in \mathrm{Thin}^{\mathrm{m}}(\Theta)} K(t; \theta) = \max_{\theta \in \Theta} K(t; \theta)$$

for $t_{\mathrm{low}} < t < t_{\mathrm{high}}$. $\mathrm{Thin}^{\mathrm{m}}(\Theta)$ can be constructed by using the thinning algorithm given above while retaining only those solutions, t, to Eqn. 14 that satisfy $t_{\mathrm{low}} < t < t_{\mathrm{high}}$. Next we define $H_n^{\mathrm{m}}(s_n, t_n)$ to be the result of applying the dynamic programming iteration of Eqn. 5 using $\mathrm{Thin}^{\mathrm{m}}$ in place of the original thinning operation. Then define $H_n^{\mathrm{c}}(s_n, t_n)$ to be the result of *constrained* optimization in which we employ Eqn. 5 but optimize only over $t_{\mathrm{low}} < t_n < t_{\mathrm{high}}$. The computation of H_n^{c} is considerably more difficult than that of H_n^{m} or H_n since the operation of "maxing out" is not so easily adapted to the constrained case, however H_n^{c} is still well-defined and will lead to the optimal constrained parse. We will show that we can, in many cases, obtain the constrained optimal parse without computing H_n^{c}.

It is easily seen by induction on n that

$$H_n^{\mathrm{m}}(s_n, t_n) \geq H_n^{\mathrm{c}}(s_n, t_n) \tag{15}$$

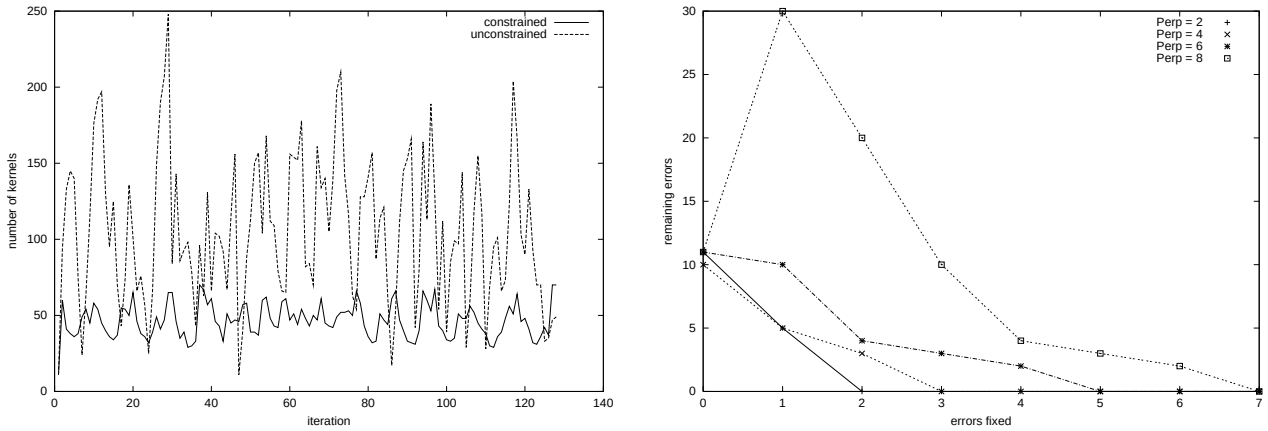

Figure 6: **Left:** The number of Gaussian kernels necessary to represent H_n, $|\Theta_n(\mathcal{S})|$, as a function of n. **Right:** The number of errors produced by our system at different perplexities and with different numbers of errors already corrected.

for $t_{\text{low}} < t_n < t_{\text{high}}$, since $H_n^{\text{c}}(s_n, t_n)$ is achieved by optimizing over a *subset* of the real line, rather than the entire real line, and the modified thinning operation, Thin^{m}, does not affect the values of $H_n^{\text{m}}(s_n, t_n)$ inside $(t_{\text{low}}, t_{\text{high}})$. However, if we construct $(\hat{s}_0^N, \hat{t}_1^N)$ using the algorithm of Section 2.2 using the *modified* thinning procedure and find that $t_{\text{low}} < \hat{t}_n < t_{\text{high}}$ for $n = 1, \ldots, N$, then we have

$$H_N^{\text{m}}(\hat{s}_N, \hat{t}_N) = L_N(\hat{s}_0^N, \hat{t}_1^N) = H_N^{\text{c}}(\hat{s}_N, \hat{t}_N)$$

The first equality is immediate; the second inequality follows since, because $t_{\text{low}} < \hat{t}_n < t_{\text{high}}$, the value $L_N(\hat{s}_0^N, \hat{t}_1^N)$ can clearly be achieved by constrained optimization but not surpassed due to Eqn. 15. Thus we see that $(\hat{s}_0^N, \hat{t}_1^N)$ is the *constrained* optimal solution since

$$H_N^{\text{c}}(\hat{s}_N, \hat{t}_N) = H_N^{\text{m}}(\hat{s}_N, \hat{t}_N) \geq H_N^{\text{m}}(s_N, t_N) \geq H_N^{\text{c}}(s_N, t_N)$$

for $t_{\text{low}} < t_N < t_{\text{high}}$ and any s_N.

While there is no guarantee that the condition $t_{\text{low}} < \hat{t}_n < t_{\text{high}}$ will hold for $n = 1, \ldots, N$, the condition was satisfied in nearly all of the experiments we have performed. Furthermore, it seems reasonable to expect that a solution $(\hat{s}_0^N, \hat{t}_1^N)$ constructed as above that nearly satisfies $t_{\text{low}} < t_n < t_{\text{high}}$ is nearly a constrained optimal solution. The computational advantage seeking constrained optima is demonstrated in the following section.

3 Experiments

We performed several experiments using two different data sets, one derived from audio data and the other taken from a MIDI performance.

3.1 Schumann Romance Data

The first data set is derived from a performance of the first section of Schumann's *2nd Romance for Oboe and Piano* (oboe part only), an excerpt of which is depicted in Figure 1. The original data, which can be heard at http://fafner.math.umass.edu/rhythmic_parsing, is a sampled audio signal, hence inappropriate for our experiments. Instead, we extracted a sequence of 129 note onset times from the data using the HMM methodology described in [21]. These data are also available at the above web page. In the performance of this excerpt, the tempo changes quite freely, thereby necessitating simultaneous estimation of rhythm and tempo.

Since the musical score for this excerpt was available, we extracted the complete set of possible measure positions,

$$\mathcal{S} = \left\{ \frac{0}{1}, \frac{1}{8}, \frac{1}{4}, \frac{1}{3}, \frac{3}{8}, \frac{5}{12}, \frac{15}{32}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8} \right\}$$

(The position 15/32 corresponds to a grace note which we have modeled as a 32nd note coming before the 3rd beat in 4/4 time). The most crucial parameters in our model are those that compose the transition probability matrix R. The two most extreme choices for R are the uniform transition probability matrix

$$R^{\text{unif}}(s, s') = 1/|\mathcal{S}|$$

and the matrix ideally suited to our particular recognition experiment

$$R^{\text{ideal}}(s, s') = \frac{|\{n : S_n = s, S_{n+1} = s'\}|}{|\{n : S_n = s\}|}$$

R^{ideal} is unrealistically favorable to our experiments since this choice of R is optimal for recognition purposes and incorporates information normally unavailable; R^{unif} is unrealistically pessimistic in employing no prior information whatsoever. The actual transition probability matrices used in our experiments were convex combinations of these two extremes

$$R = \alpha R^{\text{ideal}} + (1 - \alpha) R^{\text{unif}}$$

for various constants $0 < \alpha < 1$. A more intuitive description of the effect of a particular α value is the *perplexity* of the matrix it produces: $\text{Perp}(R) = 2^{H(R)}$ where $H(R)$ is the $\log_2$ entropy of the corresponding Markov chain. Roughly speaking, if a transition probability matrix has perplexity M, the corresponding Markov chain has the same amount of "indeterminacy" as one that chooses randomly from M equally likely possible successors for each state. The extreme transition probability matrices have

$$\begin{aligned} \text{Perp}(R^{\text{ideal}}) &= 1.92 \\ \text{Perp}(R^{\text{unif}}) &= 11 = |\mathcal{S}| \end{aligned}$$

In all experiments we chose our initial distribution, $I(s_0)$, to be uniform, thereby assuming that all starting measure positions are equally likely. The remaining constants, $\nu, \phi^2, \tau^2(s, s'), \rho^2(s, s')$ were chosen through experimentation. In particular, we modeled

$$\begin{aligned} \tau^2(s, s') &= \beta_1 l(s, s') \\ \rho^2(s, s') &= \beta_2 l(s, s') \end{aligned}$$

so only four values, $\nu, \phi^2, \beta_1, \beta_2$ were set by hand.

The computational feasibility of our approach relies on the representation of H_n from Eqn. 9 staying manageably small as n increases. The left panel of Figure 6 shows the evolution of $|\Theta_n(\mathcal{S})|$ for $n = 0, \ldots, 128$ with $\text{Perp}(R) = 4$. The figure shows results for both the basic algorithm presented in Section 2.3 and for the constrained version discussed in Section 2.3.1. In the latter version we constrained the tempo variables to lie in $(1, 5)$ corresponding to a range of 48 to 240 beats per minute (the composer's tempo marking was 104 beats per minute). Both versions show that the complexity of the representation of H_n does not grow as n increases. The average number of kernels used in the representation of $H_n(s_n, t_n)$, $\sum_{n=0}^{128} |\Theta_n(\mathcal{S})|/(129 \times |\mathcal{S}|)$, was 4.22 in the constrained case and 9.59 in the unconstrained case.

The rhythmic parsing problem we pose here is based solely on timing information. Even with the aid of pitch and interpretive nuance, trained musicians occasionally have difficulty parsing rhythms. For this reason, it is not terribly surprising that our parses contained errors. However, a virtue of our approach is that the parses can be incrementally improved by allowing the user to correct individual errors. These corrections are treated as constrained variables in subsequent passes through

the recognition algorithm. Due to the global nature of our recognition strategy, correcting a single error often fixes others parse errors automatically. Such a technique may well be useful in a more sophisticated music recognition system in which it is unrealistic to hope to achieve the necessary degree of accuracy without the aid of a human guide. In Figure 6 we show the number of errors produced under various experimental conditions. The four traces in the plot correspond to perplexities $2, 4, 6, 8$, while each individual trace gives the number of errors produced by the recognition after correcting $0, \ldots, 7$ errors. In each pass the first error found from the previous pass was corrected. In each case we were able to achieve a perfect parse after correcting 7 or fewer errors. Figure 6 also demonstrates that recognition accuracy improves with decreasing perplexity, thus showing that significant benefit results from using a transition probability matrix well-suited to the actual test data.

The experiments depicted in Figure 6 were performed with the $\{t_n\}$ constrained to line in $(1, 5)$ using the constrained thinning algorithm of Section 2.3.1. Over all experiments two of the $129 \times 8 \times 4 = 4128$ estimated tempo variables were slightly outside this range. Thus, all but two of our parses are exact constrained MAP estimates, while the other two are likely very good approximations.

3.2 Chopin Mazurka Data

In our next, and considerably more ambitious, example we parsed a MIDI performance of the Chopin Mazurka Op. 6, no. 3. for solo piano. Unlike the monophonic instrument of the previous example, the piano can play several notes at a single score position. Thus simultaneous notes, corresponding to transitions of the form $S_n = S_{n+1}$, are possible (and occur frequently).

For this example, in 3/4 time, we took the possible measure positions from the actual score, giving the set

$$\mathcal{S} = \left\{ \frac{0}{1}, \frac{1}{24}, \frac{1}{12}, \frac{1}{9}, \frac{1}{6}, \frac{2}{9}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{13}{24}, \frac{7}{12}, \frac{2}{3}, \frac{5}{6}, \frac{11}{12}, \frac{23}{24} \right\}$$

Again, several of the measure positions correspond to grace notes. Rather than fixing the parameters of our model by hand, we instead estimated them from actual data. The transition probability matrix, R, was estimated from scores of several different Chopin Mazurkas by simply counting transitions in the data and smoothing the resulting conditional distributions. The result was a transition probability matrix having $\mathrm{Perp}(R) = 2.02$, thereby providing a model that has greatly improved predictive power over the uniform transition model having perplexity $\mathrm{Perp}(R) = |\mathcal{S}| = 15$.

We also learned the variances of our model, $\tau^2(s, s')$ and $\rho^2(s, s')$ by using a different MIDI Mazurka performance with *known* score, thereby "clamping" the variables $S_0, \ldots, S_N$ to known values. Once the discrete variables are fixed, the model consists entirely of Gaussian variables and familiar techniques such as the Kalman Smoother or methods from Bayesian networks can be used to estimate posterior distributions on the $\{\delta_n\}$ and $\{\epsilon_n\}$ variables given the observed data $y_1, \ldots, y_N$. We used the EM algorithm to iterate back and forth between the computation of these posterior distribution and the reestimation of the desired variances. To smooth our estimates we used the modeling assumptions

$$\begin{aligned} \tau^2(s, s') &= \tau^2(Q(l(s, s'))) \\ \rho^2(s, s') &= \rho^2(Q(l(s, s'))) \end{aligned}$$

where Q is a function that quantizes the transition lengths into a small number of categories.

In addition, we used a slight variation on the model presented in Section 2.1 that includes the MIDI pitch of each note as an observable variable. In particular, we assume that, given the measure position, S_n, the MIDI pitch for the nth note is conditionally independent of all other variables in the model. In doing so our intention was to capitalize on the relationship between measure position and pitch. For instance, in a Mazurka the beginning of a measure is usually marked by a low note. The conditional distributions of pitch given measure position were learned from several Mazurka scores by "binning" both pitch and measure position into a small number of categories and smoothing empirical distributions.

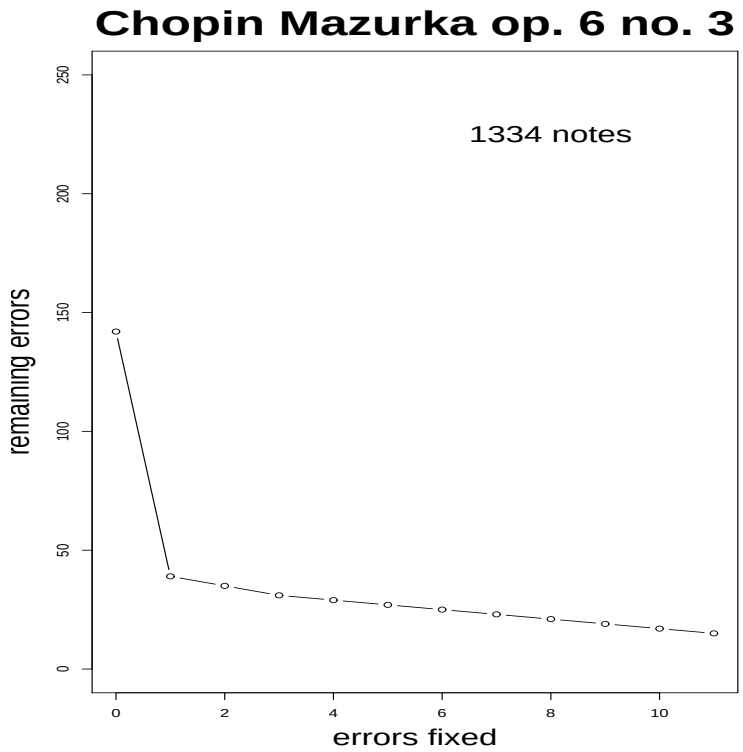

Figure 7: Results of rhythmic parses of Chopin Mazurka Op. 6, No. 3.

With this extended and automatically trained model we then iterated the procedure of parsing the data and then fixing the error beginning the longest run of consecutive errors. The results of our experiments with this 1334-note data set are shown in Figure 7. The actual MIDI performance can be heard at http://fafner.math.umass.edu/rhythmic_parsing. We see that after only a couple of corrections our error rate is in the 2-3% range. We remark that this error rate is slightly misleading since the arbitrary rhythmic notation of grace notes render the ground truth somewhat arbitrary. Many of the "errors" occurred with grace notes.

4 Discussion

We have presented a method for simultaneous estimation of rhythm and tempo, given a sequence of note onset times. Our method assumes that the collection of possible measure positions is given in advance. We believe this assumption is a relatively simple way of limiting the complexity of the recognized rhythm produced by the algorithm. When arbitrary rhythmic complexity is allowed without penalty, one can always find a rhythm with an arbitrarily accurate match to the observed time sequence. Thus, we expect that any approach to rhythm recognition will need some way to limit or penalize rhythmic complexity.

Other than the collection of possible measure positions, all parameters of our model can, and should, be learned from actual data, as in Section 3.2. Our experience is that the R matrix, which represents our prior assumptions about rhythm sequences, contains the most important parameters of our model. The estimation of this matrix requires a set of training data that "matches" the rhythmic content of the test data. For example, we would not expect successful results if we trained our model using various 4/4 time movements from Beethoven's piano sonatas and recognized on Madonna's *Material Girl*. In our experiments with the Chopin Mazurka in Section 3.2, our training data was quite well-matched to the test data, being examples of the same genre by the same composer. It is likely that a much less precise match between training and test would still prove workable. Another

possibility is to estimate the model parameters "on-line" — however, our initial experiments in this direction have not produced a significant benefit.

In our experiments it was possible to perform the dynamic programming calculations with a representation that remained bounded in complexity as described in left panel of Figure 6. We note here that this behavior is not guaranteed; if the kernels have very small variance or are highly dispersed, the thinning procedure might not produce the decrease in the complexity of the $\{H_n\}$ necessary to make the calculations feasible. However, we anticipate that the behavior we observed will be the rule rather than the exception.

The experiments presented here deal with estimating the *composite* rhythm obtained by superimposing the various parts on one another. A disadvantage of this approach is that composite rhythms can be quite complicated even when the individual voices have simple repetitive rhythmic structure. For instance, consider a case in which one voice uses triple subdivisions while another use duple subdivisions. A possible extension is the simultaneous estimation of rhythm, tempo and voicing. That is, one could model the observed data as a superposition of several independent rhythmic sources and seek to separate the sources, as well as recognize rhythm and tempo. Rhythm and voicing collective constitute the "lion's share" of what one needs for for automatic transcription of MIDI data.

5 Generalization

While the methodology in Section 2 was developed for the particular graphical model of Figure 2, the ideas extend to arbitrary graphical models for conditional Gaussian distributions. While a complete description of this generalization is beyond the scope of this paper, we sketch here such an extension. A complete description of this work can be found in [22].

A mixed collection of discrete and continuous variables, X, has a conditional Gaussian (CG) distribution if, for every configuration of the discrete variables, the conditional distribution on the continuous variables is multivariate Gaussian [14], [15]. We assume we have a representation of the CG distribution in terms of a DAG in which discrete nodes have no continuous parents.

If some of the components of X are observed, we can factor the conditional density on the remaining unobserved variables, X_U, as

$$\bar{f}(x_U) = \prod_{C \in \mathcal{C}} \phi_C(x_C) \tag{16}$$

where $\mathcal{C}$ are the cliques of a junction tree and the potential functions, $\phi_C(x_C)$, depend only on the indicated variables. When C contains continuous variables, ϕ_C can be shown to be of the form

$$\phi_C(x_C) = K(x_{\Gamma(C)}; \theta(x_{\Delta(C)})) \tag{17}$$

where $\Gamma(C)$ and $\Delta(C)$ index the continuous and discrete variables of C. Otherwise ϕ_C is the usual discrete potential.

The idea of dynamic programming can be extended beyond the linear graph structure encountered in Section 2, to maximize a function of the form of Eqn. 16 with clique potentials as in Eqn. 17. In this context, we define

$$H_C(x_C) = \max_{x_{U \setminus C}} \prod_{\tilde{C} \leq C} \phi_{\tilde{C}}(x_{\tilde{C}})$$

where $\tilde{C} < C$ if C lies on the unique path between $\tilde{C}$ and the root of the junction tree, (our tree grows downward). Then H_C can be computed recursively by the dynamic programming iteration

$$H_C(x_C) = \phi_C(x_C) \prod_{C \xrightarrow{S} \tilde{C}} \max_{x_{\tilde{C} \setminus S}} H_{\tilde{C}}(x_{\tilde{C}}) \tag{18}$$

where we take $C \xrightarrow{S} \tilde{C}$ to mean that C and $\tilde{C}$ are neighboring cliques separated by S with $\tilde{C} < C$.

As in Section 2, a specific functional form can be used to represent the H_C functions throughout the dynamic programming recursion. Suppose C has continuous components and consider the form

$$H_C(x_C) = \max_{\theta \in \Theta(x_{\Delta(C)})} K(x_{\Gamma(C)}; \theta) \tag{19}$$

(H_C is just a non-negative function when x_C has only discrete components). The terminal cliques clearly have H_C of this form, where the maximum has a single Gaussian kernel. Furthermore one can show that if all child cliques of C have such a representation, then the Eqn. 18 also leads to a similar representation for H_C. Having computed the H_{C_r} for the root clique, C_r we can easily trace back the calculations to find the optimal configuration $\hat{x}_U$.

While most of the methodology presented in Section 2 extends in a straightforward manner to the general domain of CG distributions, there is one notable exception. The computation of Eqn. 19 also involves the thinning operation of Eqn. 8, however, the collection of kernels we consider are not necessarily one-dimensional. Thus, the algorithm of Section 2.3, which is inherently one-dimensional, cannot be applied. We do anticipate that a smarter algorithm can be used to compute, or at least approximate, the thinning operation in higher dimensions. The development of such an algorithm is the only missing link between our proposed methodology and a fully general approach to finding MAP estimates for unobserved variables in CG distributions.

Appendix

A Gaussian kernel is a multivariate function of the form of Eqn. 2. The following identities hold for such functions. The derivations of these results are quite straightforward and are not included here.

Multiplication:

$$K(x; h_1, m_1, Q_1)K(x; h_2, m_2, Q_2) = K(x; h, m, Q) \tag{20}$$

where

$$\begin{aligned} h &= h_1 h_2 e^{-\frac{1}{2}(m_1^t Q_1 m_1 + m_2^t Q_2 m_2 - m^t Q m)} \\ m &= Q^-(Q_1 m_1 + Q_2 m_2) \\ Q &= Q_1 + Q_2 \end{aligned}$$

where Q^- is the *generalized inverse* of Q [23], [24]. We deal here only with nonnegative definite symmetric matrices; in this case Q^- can be expressed as

$$Q^- = UD^-U^t$$

where $Q = UDU^t$ with U unitary and D diagonal, and D^- is diagonal with

$$D^-_{ii} = \begin{cases} 1/D_{ii} & D_{ii} > 0 \\ 0 & D_{ii} = 0 \end{cases}$$

Maxing Out: Let m and Q be partitioned as

$$m = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \tag{21}$$

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \tag{22}$$

Then

$$\max_{x_2} K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) = K(x_1; h, m_1, \tilde{Q}) \tag{23}$$

where

$$\tilde{Q} = Q_{11} - Q_{12}Q_{22}^-Q_{21}$$

In the event that x_1 has *no* components, we have maximized over all variables of the kernel and interpret $K(x_1; h, m_1, \tilde{Q})$ as the constant h.

Fixing Variables: Let m and Q be partitioned as in Eqns. 21,22. Regarding x_2 as fixed

$$K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) = K(x_1; \tilde{h}, \tilde{m}, \tilde{Q}) \tag{24}$$

where

$$\begin{aligned} \tilde{h} &= h e^{-\frac{1}{2}(x_2 - m_2)^t (Q_{22} - Q_{21} Q_{11}^- Q_{12})(x_2 - m_2)} \\ \tilde{m} &= m_1 - Q_{11}^- Q_{12}(x_2 - m_2) \\ \tilde{Q} &= Q_{11} \end{aligned}$$

Extension: The kernel $K(x_1; h, m, Q)$ can be viewed as a function of x_1 and x_2 by

$$K(x_1; h, m, Q) = K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, \begin{pmatrix} m \\ 0 \end{pmatrix}, \begin{pmatrix} Q & 0 \\ 0 & 0 \end{pmatrix}) \tag{25}$$

Conditional Gaussian Densities: If $x_2 = \alpha^t x_1 + \beta + \xi$ where x_2 is univariate and $\xi \sim N(\mu, \sigma^2)$, the conditional density of x_2 given x_1 is

$$f_v(x_2 | x_1) = K(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; h, m, Q) \tag{26}$$

where

$$\begin{aligned} h &= (2\pi\sigma^2)^{-1/2} \\ m &= \begin{pmatrix} 0 \\ \beta + \mu \end{pmatrix} \\ Q &= \frac{1}{\sigma^2} \begin{pmatrix} \alpha\alpha^t & -\alpha \\ -\alpha^t & 1 \end{pmatrix} \end{aligned}$$

References

[1] Desain P., Honing H. (1989), "The Quantization of Musical Time: A Connectionist Approach," *Computer Music Journal*, Vol 13, no. 3.

[2] Trilsbeek P., van Thienen H., (1999), "Quantization for Notation: Methods used in Commercial Music Software," handout at *106th Audio Engineering Society conference*, May 1999, Munich.

[3] Cemgil A. T., Kappen B., Desain P., Honing, H. (2001), "On Tempo Tracking: Tempogram Representation and Kalman Filtering" In Press at *Journal of New Music Research*

[4] Desain P., Honing H. (1994), "Foot Tapping: A Brief Introduction to Beat Induction," *Proceedings of the International Computer Music Conference*, 78–79, International Computer Music Association, San Francisco CA, 1994.

[5] Dixon S., (2000), "A Lightweight Multi-Agent Musical Beat Tracking System," *Proceedings of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance, and Analysis*, Austin TX, July 2000.

[6] Dixon S., (1999), "A Beat Tracking System for Audio Signals," *Proceedings of the Diderot Forum on Mathematics and Music*, Austrian Computer Society.

[7] Allen P., Dannenberg R. (1990), "Tracking Musical Beats in Real Time," In *Proceedings of the International Computer Music Conference*, 140–143, International Computer Music Association, San Francisco CA, 1990.

[8] Dannenberg R., Mont-Reynaud B., (1987), "Following an Improvisation in Real Time," In *Proceedings of the International Computer Music Conference*, 241–248, International Computer Music Association, San Francisco CA, 1987.

[9] Goto M., Muraoka Y., (1995) "A Real-Time Beat Tracking System for Audio Signals," *Proceedings of the International Computer Music Conference*, 171–174, International Computer Music Association, San Francisco CA, 1995.

[10] Goto M., Muraoka Y., (1998) "An Audio-Based Real-Time Beat Tracking System and its Applications," *Proceedings of the International Computer Music Conference*, 17–20, International Computer Music Association, San Francisco CA, 1998.

[11] Desain P., Aarts R., Cemgil A. T., Kappen B., van Thienen H, Trilsbeek P. (1999), "Robust Time-Quantization for Music from Performance to Score," *Proceedings of 106th Audio Engineering Society conference*, May 1999, Munich.

[12] Cemgil A. T., Desain P., Kappen B. (1999), "Rhythm Quantization for Transcription," *Computer Music Journal*, 60-76.

[13] Cemgil A. T., (2001), "Tempo Tracking and Rhythm Quantization by Sequential Monte Carlo" to appear in *Advances in Neural Information Processing Systems 14*, MIT Press, 2002.

[14] Lauritzen S. L. and Wermuth N (1984), "Mixed Interaction Models," *Technical Report R-84-8*, Institute for Electronic Systems, Aalborg University.

[15] Lauritzen S. L. and Wermuth N (1989), "Graphical Models for Associations Between Variables, some of which are Qualitative and some Quantitative," *Annals of Statistics*, 17, 31-57.

[16] Lauritzen S. L. (1992), "Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models," *Journal of the American Statistical Association*, Vol. 87, No. 420, (Theory and Methods), 1098–1108.

[17] Lauritzen S. L. and F. Jensen (1999), "Stable Local Computation with Conditional Gaussian Distributions," *Technical Report R-99-2014*, Department of Mathematic Sciences, Aalborg University.

[18] Lerner U., Parr R., (2001), "Inference in Hybrid Networks: Theoretical Limits and Practical Algorithms," In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence,(UAI 2001)*, 310–318, Morgan Kauffman, 2001.

[19] Cowell R. G., Dawid A. P., Lauritzen S. L., Spiegelhalter D. J. (1999), "Probabilistic Networks and Expert Systems," Springer, New York, NY.

[20] Dawid A. P. (1992) "Applications of a General Propagation Algorithm for Probabilistic Expert Systems," *Statistics and Computing*, vol. 2, 25-36.

[21] Raphael C., (1999), "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no 4, 360 – 370, 1999.

[22] Raphael C. (2001), "MAP Estimation of Unobserved Variables in Conditional Gaussian Distributions," submitted to *Journal of the American Statistical Association*,

[23] Rao C. R., Mitra S. K., (1971), "Generalized Inverse of Matrices and its Applications," John Wiley and Sons, New York.

[24] Graybill, F., (1969), "Matrices with Applications in Statistics," Wadsworth International Group, Belmont, CA.