AUTOMATIC TRANSCRIPTION OF MUSIC AUDIO THROUGH CONTINUOUS PARAMETER TRACKING

Eric Nichols
Dept. of Computer Science
Indiana Univ.
epnichol@indiana.edu

Christopher Raphael
School of Informatics
Indiana Univ.
craphael@indiana.edu

ABSTRACT

We present a method for transcribing arbitrary pitched music into a piano-roll-like representation that also tracks the amplitudes of the notes over time. We develop a probabilistic model that gives the likelihood of a frame of audio data given a vector of amplitudes for the possible notes. Using an approximation of the log likelihood function, we develop an objective function that is quadratic in the time-varying amplitude variables, while also depending on the discrete piano-roll variables. We optimize this function using a variant of dynamic programming, by repeatedly growing and pruning our histories. We present results on a variety of different examples using several measures of performance including an edit-distance measure as well as a frame-by-frame measure.

1 INTRODUCTION

Polyphonic audio transcription has received considerable attention in recent years and is holds promise in MIR for its potential for automatically creating symbolic music representations from audio. Research in this area has produced a wide variety of approaches [1], [3]-[11] with significant contributions, though the problem is deeply challenging and remains open. A recurring theme in this work is that of representing a music spectrogram as a superposition of fixed (but trainable) templates modeling various note aspects. Examples of such template-based approaches are non-negative matrix representations for frames over notes [1], [10] or fundamental frequencies [6], or note-based models involving time-extent as well [7]. Our approach shares some methodology with [11], although that work uses peak detection instead of templates.

While these local model-fitting problems are difficult, perhaps even more challenging is the problem of assembling a global interpretation of the data, beyond the frame or note level. Such full transcription problems [3], [8], require the integration of more global musical knowledge.

Our work is in this latter category. In addition to attempting polyphonic transcription from unknown sources, we simultaneously estimate the time-varying note amplitude parameters, hoping their knowledge will lead to more discriminating models. This work has overlap with [4], though our approach is note-based, rather than harmonic-based. The amplitude envelopes themselves may be of primary interest for some applications.

2 A PROBABILISTIC MODEL

We present here a probabilistic model describing the likelihood of a frequency spectrum given an assumed configuration of sounding pitches.

We denote our sampled time signal as $x(m)$ for $m = 0, \ldots M-1$. Our entire analysis is based on the spectrogram of the time data which we define as

$$s_t(k) = \left| \sum_{n=0}^{N-1} x(tL+n) w(n) e^{\frac{2\pi i k n}{N}} \right|^2$$

where N is the frame length, L is the "hop size," w is an N-point window function, and $k = 0, \ldots, N/2$, $t = 0, \ldots, T-1$, where $T = 1 + \lfloor (M-N)/L \rfloor$.

We probabilistically model the "time slices" of the spectrogram, $s_t(\cdot)$, as follows. Suppose that we begin with an idealized template spectrum, $q(i, \cdot)$, for each possible note indexed by $i = 1, \ldots I$. Here we assume that $q(i, \cdot)$ is a probability distribution so that $q(i,k) \geq 0$ and $\sum_k q(i,k) = 1$. These note models can either be estimated from actual data, as discussed in a later section, or simply fixed according to a predetermined model. In this latter case we have used the mixture of discrete Gaussians model

$$q(i,k) = \sum_{h=1}^{H} p_h N(k; \mu(i,h), \sigma^2(i,h)) \tag{1}$$

where $N(k; \mu, \sigma^2) = P(k - 1/2 < X < k + 1/2)$ for normally distributed X with mean $\mu(i,h) = h\omega_0(i)$ and variance $\sigma^2(i,h) = ah\omega_0(i) + b$. Here $\sum_h p_h = 1$ and $\omega_0(i)$ is the fundamental frequency of the ith pitch.

Let $\alpha_t(i) \geq 0$ denote the contribution of pitch i during frame t, so that $\alpha_t(i) = 0$ if the note is absent and

$$q_{\alpha_t}(k) = \sum_i \alpha_t(i) q(i,k) \tag{2}$$

denotes the idealized template for frame t. We model the data likelihood at frame t, given $q_{\alpha_t}(\cdot)$, by assuming that the $\{s_t(k)\}$ are independent and that $s_t(k) \sim$

$\text{Poisson}(q_{\alpha_t}(k))$. For this assumption to make sense, we scale to a point where the truncation to integral values produces no significant loss. The log likelihood can then be written as

$$\begin{aligned}\log \quad P \quad & (s(t,\cdot)|q_{\alpha_t}(\cdot)) & (3)\\ = \quad & c + \sum_{k=0}^{N/2} s(t,k)\log q_{\alpha_t}(k) - q_{\alpha_t}(k) & (4)\\ = \quad & c + \sum_{k=0}^{N/2} s_t(k)\log\left(\sum_{i=1}^{I}\alpha_t(i)q(i,k)\right)\\ & -\sum_{i=1}^{I}\alpha_t(i)q(i,k)\end{aligned}$$

where c is a constant not depending on q_{α_t}. Thus we have a parametric probability model with parameters $\{\alpha_t(i)\}$.

3 BUILDING THE LATTICE

Our immediate goal is to create a collection of hypotheses for the notes that sound in each frame. For simplicity, we will call these collections "chords" while acknowledging that they are not exactly the same as the usual musical meaning of the word. The collections of chord hypotheses, indexed by the frame t, can be thought of as a lattice. Our audio recognition will be phrased as a search for the best path through this lattice. Since a hypothesis that is missing from this lattice will never be considered during recognition, we wish to err on the side of *inclusion* by admitting any hypothesis that seems plausible.

Our search for candidate hypotheses is made easier by the tractable nature of our log likelihood function. Computing derivatives of the log likelihood with respect to the α parameters shows that the negative of the Hessian of the log likelihood is nonnegative definite. Thus the log likelihood is convex in the α parameters. The virtue here is that the numerical optimization of the log likelihood over the α parameters is relatively easy to perform while we are assured of finding a global optimum due to convexity. The only minor difficulty is handling the positivity constraints on the α parameters. We have accomplished this using the barrier method, [2], which approximates the constrained optimization by a series of unconstrained optimization problems.

Let $\alpha_t^*(i)$ be the optimal parameters found by maximizing the log likelihood for frame t. We create our initial set of chord hypotheses for frame t by taking up to R notes whose contribution to the overall spectrum exceeds a threshold.

$$S(t) = \{i \in I : \alpha_t^*(i) \geq \max(T_{\min}, \alpha_t^*(i_{(R)}))\}$$

where $\alpha_t^*(i_{(R)})$ is the Rth highest of the $\{\alpha_t^*(i)\}$ ($R = 5$ in our experiments). Our initial chord hypotheses for frame t are then taken to be the *power set*, or set of all possible $2^{|S(t)|}$ subsets, of $S(t)$. We denote this initial collection of hypotheses by $L(t) = \mathcal{P}(S(t))$. As mentioned before, we hope not to miss possible hypotheses at this stage so we try to be as liberal as possible in the construction of our initial lattice. To this end, we form an expanded lattice including all chords detected in nearby frames as well.

4 A SIMPLE RECOGNITION SCHEME

We wish to label each frame, t, with a chord label, $C(t) \in \bar{L}(t)$ such that the data support each chord label and the *horizontal* evolution of the chords makes a certain minimal musical sense. To this end, each sequence of frames $C = (C_0, \ldots C_{T-1})$ is scored as $S(C) = D(C) + E(C)$ where the data score, $D(C)$, and the penalty term, $E(C)$, measure these two aspects of the hypothesis C.

The data score is defined by

$$D(C) = \sum_{t=0}^{T-1} \log P(s_t|q_{\alpha_{C_t}})$$

where

$$\alpha_{C_t}(i) = \begin{cases} \sum_k s_t(k)/|C_t| & i \in C_t\\ 0 & \text{otherwise}\end{cases}$$

In other words, we use a template created by equal contributions of the hypothesized notes such that the total energy explained by the template is equal to the total energy in the spectrum. The penalty term is given by

$$E(C) = -L_{\text{enter}} \sum_{t=1}^{T-1} |\{i \in C(t) : i \notin C(t-1)\}|$$

where L_{enter} gives the penalty for each "entering" note.

With this penalty function it is a simple matter to find the globally optimal path through the lattice using dynamic programming.

5 SIMULTANEOUS RECOGNITION AND AMPLITUDE TRACKING

The recognition scheme presented in Section 4 suffers from some weaknesses. For one, it is not possible to recognize rearticulations (repeated pitches) since the data term is identical for both held and rearticulated hypotheses, while the penalty term will always prefer the hypothesis with fewer notes. In addition, the model assumes that the amplitudes (α's) of all of the notes in a "chord" are the same. Given no other information, this seems as reasonable as any assumption, but is certain to be far from reality. We introduce in this section a method for simultaneous recognition of the chord sequence and tracking of the amplitude parameters. The method has much the same flavor as Rao-Blackwellized particle filtering, though we seek a most-likely path, rather than a filtered solution.

Our essential approach performs dynamic programming to find the best chord sequence, as before. However, each chord hypothesis is "scored" not by a number, but by a *function* that measures the goodness of the hypothesis as a function of the unknown amplitude parameters.

This function is represented parametrically, so that we can update the parameters as the hypothesis ages — as in the Kalman filter. This method overcomes some weaknesses of the previous approach by representing the data likelihood as a function of the amplitude parameters — clearly a desirable trait for the data model.

A complete hypothesis for our data is now given as a sequence of chords, $C = (C_0, \ldots, C_{T-1})$, along with the amplitude vectors, $\alpha = (\alpha_0, \ldots, \alpha_{T-1})$, and a binary breakpoint vector $b = (b_0, \ldots, b_{T-1})$ with $b_0 = 0$ and $b_{T-1} = 0$. We require the chord to be constant between breakpoints: $C_t = C_{t-1}$ for $b_t \neq 0$, though we do not require the chord to change after a breakpoint. In this way we represent something like a rearticulation, though we do not distinguish between the rearticulating and sustaining chord members. Each amplitude vector describes the amplitudes of the notes in the current chord with the non-chord members constrained to have 0 amplitude: $\alpha_t(i) = 0$ for $i \notin C_t$. In representing the "goodness" of a hypothesis we penalize both hypotheses that don't agree with the data and hypotheses that are not musically plausible.

In our new version, the data score depends on the unknown amplitude parameters (the α_t parameters of Eqn. 3). Since this log likelihood function was shown to be convex in the amplitude parameters in Section 3, it seems reasonable to approximate it by a quadratic function in the $\{\alpha(t)\}$ parameters. We parameterize this by

$$\log P(s_t|q_{\alpha_t}) \approx h_t + \frac{1}{2}(\alpha_t - m_t)^t Q_t (\alpha_t - m_t)$$

where (h_t, m_t, Q_t) are found by completing the square on the Taylor series approximation of the data log likelihood expanded around our optimizing point, α_t^*. We don't know about the non-local quality of the quadratic approximation, but this approximation is quite useful for the computations that follow. For instance, if we want to approximate the data log likelihood for a specific chord, C_t, as a function of the amplitude parameters, we need only restrict to zero the $\alpha_t(i)$ parameters where $i \notin C_t$. We will denote this restriction by $(h_{C_t}, m_{C_t}, Q_{C_t})$. Here Q_{C_t} is obtained simply by zeroing out the rows and columns of Q_t not contained in $C(t)$, while $h_C(t)$ and $m_C(t)$ are obtained as simple linear functions of the original parameters (as in the Gaussian conditional mean). The data score of a hypotheses is then

$$\begin{aligned} D(C, \alpha) &= \sum_{t=0}^{T-1} D_t(C_t, \alpha_t) \\ &\overset{\text{def}}{=} \sum_{t=0}^{T-1} \frac{1}{2}(\alpha_t - m_{C_t})^t Q_{C_t} (\alpha_t - m_{C_t}) \end{aligned}$$

We modify our previous penalty term by adding in a component that favors α_t vectors that vary smoothly over the duration of a chord, as would be expected. That is

$$\begin{aligned} E(C, b, \alpha) &= \sum_{t=0}^{T-1} E_t(C_{t-1}, C_t, b_t, \alpha_{t-1}, \alpha_t) \\ &\overset{\text{def}}{=} -\sum_{b(t)=1} L_{\text{enter}} |C_t - C_{t-1}| \\ &\quad - \sum_{b(t)=0} r||\alpha_t - \alpha_{t-1}||^2 \end{aligned}$$

for some constant $r > 0$. Note that we have no reasonable priors on the parameters α_t necessary for a Bayesian formulation of this problem; instead, we prefer the framework of maximum likelihood, with the penalty term providing smoothing. Our goal is now to maximize our total hypothesis score, $S(C, b, \alpha) = D(C, \alpha) + E(C, b, \alpha)$.

5.1 Performing the Optimization

Our score function is more difficult to optimize than that of Section 4 due to the combination of discrete (C, b) and continuous (α) parameters. Still, our basic approach is to perform dynamic programming just as before. To that end we use the vector notation $a_0^t = (a_0, \ldots, a_t)$ for any vector a, and define

$$\begin{aligned} S_t(C_0^t, b_0^t, \alpha_0^t) &= \sum_{\tau=0}^{t} D_\tau(C_\tau, \alpha_\tau) \\ &\quad + E_\tau(C_\tau, C_{\tau-1}, b_\tau, \alpha_{\tau-1}, \alpha_\tau) \end{aligned}$$

which can be written more compactly with the notation $x_t = (C_t, b_t, \alpha_t)$ as

$$S_t(x_0^t) = \sum_{\tau=0}^{t} D_\tau(x_\tau) + E_\tau(x_{\tau-1}, x_\tau)$$

Our dynamic programming recursion is then

$$S_t^*(x_t) = \max_{x_{t-1}} S_{t-1}^*(x_{t-1}) + D_t(x_t) + E_t(x_{t-1}, x_t) \quad (5)$$

where $S_t^*(x_t)$ is the score of the optimal partial hypothesis ending in x_t.

Unfortunately, the dynamic recursion cannot be computed in the usual way, due to the continuous parameters in Eqn. 5. However, we can approximate these calculations. Our approach is to grow a tree of partial hypotheses where a node in the tree at depth t has associated discrete variables b_t, C_t where $C_t \in L(t)$. After the tth iteration our algorithm, the terminal nodes are all at depth t, so each path from the root to the terminal node corresponds to a possible history of the discrete variables. Rather than scoring each terminal node with a single number, we score the terminal nodes with a function that measures the "goodness" of a history as a function of the unknown continuous variables at frame t. We will alternately grow and prune the tree in an effort to approximate the dynamic programming recursion.

Suppose that C_0^t, b_0^t is a possible discrete history. The function describing the quality of this hypothesis is

$$\begin{aligned} \tilde{S}_{C_0^t, b_0^t}(\alpha_t) &= \max_{\alpha_0^{t-1}} S(C_0^t, b_0^t, \alpha_0^t) \quad (6) \\ &= h_{C_0^t, b_0^t} \quad (7) \\ &\quad + (\alpha_t - m_{C_0^t, b_0^t})' Q_{C_0^t, b_0^t} (\alpha_t - m_{C_0^t, b_0^t}) \end{aligned}$$

where the maximum is over α_0^{t-1} such that $\alpha_\tau(i) = 0$ when $i \notin C_\tau(i)$ for $\tau = 0, \ldots, t-1$. This function can be computed recursively, using the usual dynamic programming idea, due to the quadratic dependence on the continuous parameters.

Thus, each hypothesis concerning the first t frames corresponds to a terminal node in our tree at level t, corresponding to the discrete history C_0^t, b_0^t and the parametrically represented score function of Eqns. 6,7.

5.2 Pruning the Hypotheses

Our tree construction will not be feasible without some pruning of the histories. Our approach to pruning is based on the following observation. Suppose we have two discrete histories, C_0^t, b_0^t and $\bar{C}_0^t, \bar{b}_0^t$ where $C_t = \bar{C}_t$ and $b_t = \bar{b}_t$. If $\tilde{S}_{C_0^t, b_0^t}(\alpha_t) > \tilde{S}_{\bar{C}_0^t, \bar{b}_0^t}(\alpha_t)$ for all *legal* α_t — that is, α_t with non-negative coordinates such that $\alpha_t(i) = 0$ when $i \notin C_t$ — then any continuation of $\bar{C}_0^t, \bar{b}_0^t$ will always score worse then the same continuation of C_0^t, b_0^t. Thus, the history $\bar{C}_0^t, \bar{b}_0^t$ may be pruned with no possibility of losing the optimal history in our tree construction. We call such a pruning a "dp cutoff." This idea extends to the case in which a collection of hypotheses "dominate" another hypothesis, though the determination of such a collection is computationally challenging.

We approximate this idea by performing dp cutoffs when a discrete hypothesis seems *unlikely* to be optimal for any legal value of α_t. We do this by approximating, for each surviving hypothesis, $\bar{C}_0^t, \bar{b}_0^t$, the legal point α_t^* where $\tilde{S}_{\bar{C}_0^t, \bar{b}_0^t}$ is maximum. If, for this point

$$\tilde{S}_{C_0^t, b_0^t}(\alpha_t^*) > \tilde{S}_{\bar{C}_0^t, \bar{b}_0^t}(\alpha_t^*)$$

for *any* of the surviving C_0^t, b_0^t we conclude that, having failed to be maximal at its maximizing point α_t^*, $\tilde{S}_{\bar{C}_0^t, \bar{b}_0^t}$ is unlikely to be maximal at any legal point α_t. Thus, we prune $\bar{C}_0^t, \bar{b}_0^t$.

Our tree construction then proceeds as follows. After each iteration, t, we have a collection of histories, H_t, each with a quadratic score function as in Eqns. 6,7. In the $t+1$ iteration, we produce the set

$$\begin{aligned} \tilde{H}(t+1) \quad = \quad & \{(C_0^{t+1}, b_0^{t+1}) : (C_0^t, b_0^t) \in H(t), \\ & C_{t+1} \in L(t+1), \\ & b_{t+1} = 0 \text{ if } C_{t+1} \neq C_t\} \end{aligned}$$

Thus every surviving history can be continued by any possible frame hypothesis in $L(t+1)$, while we only allow the possibility of rearticulation $b_{t+1} = 1$ for a continued chord. For each new history in $\tilde{H}(t+1)$ we compute the optimal score function $\tilde{S}_{C_0^{t+1}, b_0^{t+1}}(\alpha_{t+1})$. On these histories we first prune according to the procedure described above for dynamic programming cutoffs. This phase approximates the process of pruning histories that could not be part of the eventual optimal history. A second pruning phase further discards histories until we are left with a fixed number of possibilities. In this phase the histories are sorted according to our approximation of the best score they could produce on the legal space. We then retain the M best histories and denote this set by H_{t+1}.

6 TRAINING THE MODEL

Some of our experiments have been performed using the reasonable, but somewhat arbitrary, probability model for the audio spectrum given in Eqn. 1. The advantage of such an approach is that it can be applied to any music audio, without prior knowledge of the many factors leading to the presentation of a note spectrum. However, given such prior knowledge, we should be able to improve our model to better capture reality.

We assume that our audio training data is accompanied with both a symbolic representation of the audio, as in a MIDI file, as well as a *score match*, giving a correspondence between the symbolic representation and the audio. We assume only one model for every possible pitch, as would be appropriate for piano music.

As discussed in section 2, we model the $\{s_t(k)\}$ as realizations of independent Poisson random variables with

$$E s_t(k) = q_{\alpha_t}(k) = \sum_{i \in C_t} \alpha_t(i) q(i,k)$$

where, during the training phase, C_t is *known* due to the score match. This assumption can be viewed as a consequence of the assumption that $s_t(k) = \sum_{i \in C_t} Z_t(i,k)$ where the $\{Z_t(i,k)\}$ are independent Poisson random variables with $E Z_t(i,k) = \alpha_t(i) q(i,k)$. We view the training process as maximum likelihood estimation of the $\{\alpha_t(i)\}$ and $\{q(i,k)\}$ parameters. Though we must estimate both of these together, we are only interested in the "template" spectra $q(i, \cdot)$ where we continue to assume that $\sum_k q(i,k) = 1$. These template spectra and $\alpha_t(i)$ parameters are estimated using a straightforward application of the EM algorithm in which the $Z_t(i,k)$ are regarded as the hidden variables while $s_t(k)$ is observable.

7 EXPERIMENTS

We implemented and tested the two algorithms described above on several diverse musical examples:

- Bach WTC I, C minor fugue (performed on piano)
- Beethoven Duet No. 3 for Clarinet and Bassoon
- Brahms Symphony 2, Mvt. 1
- Copland Fanfare for the Common Man
- Haydn String Quartet No. 62 ("Emperor")

We chose minute-long excerpts from recordings of each piece and sampled down to 8 kHz mono audio. The frame size was fixed to 512 samples with a hop size of 256 samples for all experiments.

Evaluation of transcription results is itself a nontrivial task. Here we adopt a dual strategy of providing both quantitative accuracy results and a qualitative discussion of the algorithm output.

		L_{enter} L_{enter} / r	Mean Correct Notes/Frame	Mean Expected Notes/Frame	Percent Correct	Mean Extra Notes/Frame	Edit Dist % cor–ins–del
Bach	simple	0.1	1.6/1.9/1.5	2.4/2.2/2.1	70/85/73	1.7/1.2/0.7	59–41–41
	simple EM	0.1	1.8/1.8/1.5	"/"/"	78/83/71	1.0/0.7/0.5	64–36–21
	continuous	0.1/100	1.4/1.5/1.4	"/"/"	58/70/68	1.6/1.2/0.9	42–58–14
Beethoven	simple	0.5	1.4/1.4/1.2	2.0/1.8/1.8	70/79/64	1.1/0.7/0.5	51–49–42
	continuous	0.2/50	0.1/0.3/0.1	"/"/"	6/16/63	2.8/2.2/1.2	20–80–68
Brahms	simple	0.4	1.8/2.5/1.3	5.6/3.5/2.1	33/73/62	1.3/0.8/0.6	18–82–15
	continuous	0.4/200	0.8/1.4/0.9	"/"/"	14/40/44	2.0/1.3/0.9	17–83–13
Copland	simple	1.0	1.2/1.3/0.8	2.2/1.9/1.9	55/67/43	0.4/0.2/0.1	41–59–5
	simple EM	1.0	1.3/1.3/0.9	"/"/"	58/70/49	0.6/0.4/0.3	36–64–23
	continuous	0.7/200	1.1/1.3/1.0	"/"/"	48/65/52	1.0/0.5/0.4	28–72–38
Haydn	simple	0.5	2.0/2.2/1.6	3.5/2.9/2.5	55/77/65	1.1/0.7/0.5	39–61–20
	continuous	0.3/100	1.7/1.9/1.3	"/"/"	48/64/54	0.9/0.5/0.3	32–68–23

Table 1. Results: each entry is computed with three different scoring metrics: exact/pitch class/min. homonym.

7.1 Quantitative Evaluation

To evaluate the algorithms numerically, we first convert the computed optimal paths into MIDI files. For the simple recognition scheme (Section 4) we traverse the optimal path and generate a MIDI 'note on' message at any frame where a new note appears. A 'note off' is generated at the first frame where the note does not appear. We generate MIDI for the continuous tracking algorithm (Section 5) in a similar manner, but with the added condition that a note must have an amplitude greater than a small positive cutoff before being considered 'on'.

Comparing the MIDI files to the score match, we collected the accuracy results shown in Table 1.

Parameter values The values for L_{enter} used in each experiment. r is given for continuous tracking experiments.
Mean correct notes per frame The average number of correct notes generated for each audio frame. The three values in each cell correspond to three different evaluation metrics (see below).
Mean expected notes per frame The total number of notes in the target score divided by the number of frames.
Percent correct The total number of correct notes divided by expected notes, ignoring extra notes.
Mean extra notes per frame These are incorrect notes; i.e., any notes generated by the algorithm that were not listed in the target score for each frame.
Edit distance We compute a version of edit distance adapted to polyphonic music. Unlike the other metrics in the table, edit distance is unconcerned with note durations and audio frames. The three numbers in each cell are the number of correct matches, insertions, and deletions, expressed as percentages of the number of target notes. Note that due to the large number of insertions and deletions, we report these values separately for clarity instead of combining all three into a single score.

For cells with multiple metrics computed, the first entry corresponds to the natural (exact) metric: a note is correct if it appears in the target score; otherwise it is labeled as 'extra'. The second metric reported is based on pitch class: for each frame, the output chord and target chord are both reduced to a set of pitch classes (i.e. octave errors are ignored). The final metric is similar, but each chord is reduced to a 'minimal homonym' by ignoring notes that are harmonics of notes lower in pitch.

7.2 Parameter Selection and Audio Output

Both algorithms include a set of parameters that can be tuned to improve performance. The values of L_{enter} and r shown above were selected by trial and error. Values of L_{enter} between 0.1 and 1.5 typically seemed best, as did values of r ranging between 10 and 200. Other crucial parameters that remained fixed included:

Maximum number of notes per frame Fixed to five (simple algorithm) or four (continuous).
Maximum number of histories Fixed to 150; the tree of best paths was pruned to this size at each iteration.
Minimum amplitude Notes with energy less than this cutoff were ignored in the lattice generation phase. Set to 0.01 (simple) or 0.05 (continuous).

The algorithm to generate MIDI files described above has two main weaknesses. First, in cases where the same note is rearticulated without an intervening rest, no additional MIDI 'note on' will be generated. Repeated notes are thus be combined into single long MIDI notes, eliminating rearticulations from the output. Second, the MIDI files do not take advantage of the continually varying amplitude data captured by the continuous model.

We generated audio output files [1] from the continuous algorithm by superimposing sine waves. For each note in each frame of audio, we generated the fundamental frequency with its computed amplitude and added in five overtones with amplitudes decreasing as $1/n$.

[1] Audio files in .wav format are available online at `http://xavier.informatics.indiana.edu/~craphael/ismir07/transcription/`

7.3 Discussion

Both algorithms begin by generating a hypothesis lattice. For most experiments, the correct hypothesis appears in about 90% of the lattice frames, giving us an upper bound on the accuracy we can expect from later stages. Our best result was the Bach excerpt, with 78% notes exactly correct when using the simple algorithm with trained note templates. Without EM training performance drops to 70%. This excerpt is perhaps the simplest in terms of timbre for our model, because it only involves one instrument. Even when trained, our model assigns only one probability distribution to each MIDI note. In a more complex piece such as the Brahms symphony, the different timbres of multiple instruments may confound the pitch model.

Results of the pitch class and homonym metrics also point to problems related to instrument timbre: there is generally a significant jump in the percent correct when we consider the pitch class (octave equivalence) metric. For example, in the simple algorithm result for Bach, there is a 15% increase in accuracy if we ignore octave mistakes. Octave errors arise from misinterpreting the fundamental frequency, which can happen easily if the model misrepresents the true distribution of energy across different harmonics. In the Bach case, training the model reduces the disparity between the exact and pitch class metrics, suggesting a timbre problem before training.

Table 1 shows that there are often as many extra notes as correct notes. This may be the most glaring weakness of the results. However, the homonym metric sheds some light: there are far fewer extra notes per frame when we discount notes that are members of the overtone series of correct notes. Listening to the results suggests another source of errors: some notes last too long and overlap successive notes. Due to reverberation and resonance, notes persist longer than notated in a musical score. Determining the perceived cutoff time of a note, as opposed to the acoustic decay time, is a difficult open question.

Surprisingly, the simple algorithm always scored better than the continuous tracking version. However, recall that the continuous results are based on comparison of the generated MIDI file with the target output. Indeed, the audio synthesized from the continuous algorithm sounded consistently better than the MIDI output from the simple algorithm. The lifelike quality of the results suggests that continually-tracked amplitudes capture more performance details than simple note onset events. The output recalls the dynamic range of a piece, balance between instruments, and crescendos/decrescendos within a single note.

While there is no standardized collection of audio data and associated evaluation metrics in the field of polyphonic audio recognition, our results subjectively appear not to be as good as some other work cited in the introduction. However, our approach solves the slightly different problem of simultaneous note and amplitude tracking. The continuous amplitude values output by our algorithm may be useful in a transcription system that aims to capture dynamics as well as pitches, or, similarly, in a system for analysis of expressive musical performance.

8 REFERENCES

[1] Abdallah S., Plumbley M., "Polyphonic Transcription by Non-Negative Sparse Coding of Power Spectra," *Proceedings of the Fifth International Conference on Music Information Retrieval, ISMIR 2004* pp. 318-325, Barcelona, Spain, 2004.

[2] Boyd S. P., Vandenberghe L. 2004. *Convex Optimization*. Cambridge, UK: Cambridge University Press.

[3] Cemgil A. T., Kappen H. J. , Barber D. "A Generative Model for Music Transcription," *IEEE Transactions on Audio, Speech and Language Processing* 14(2), March 2006.

[4] Dubois C., and Davy M., "Joint Detection and Tracking of Time-Varying Harmonic Components: a General Bayesian Framework," *IEEE Transactions on Audio, Speech and Language Processing* to appear.

[5] Godsill S., and Davy M., "Bayesian Harmonic Models for Musical Pitch Estimation and Analysis," *Proc. IEEE International Conf. on Acoustics Speech, and Signal Processing (ICASSP2002),* pp. 1769-1772.

[6] Goto M., "A Real-Time Music Scene-Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-World Audio Signals," *ISCA Journal*, 43(4):311-329,2004.

[7] Kameoka H., Nishimoto Y., Sagayama S., "'Audio Stream Segregation of Multi-Pitch Music Signal Based on Time-Space Clustering Using Gaussian Kernel 2-Dimensional Model," *Proc. IEEE International Conf. on Acoustics Speech, and Signal Processing (ICASSP2005).*

[8] Kapanci E., Pfeffer A., "Signal-to-Score Music Transcription Using Graphical Models," *Proc. 19th Int. Joint Conf. on Artif. Intel. (IJCAI)*, Edinburgh, UK, August 2005.

[9] Klapuri A., "Multiple Fundamental Frequency Estimation by Harmonicity and Spectral Smoothness," *IEEE Trans. Speech and Audio Processing*, 11(6), 804-816, 2003.

[10] Klapuri A., "Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes," *Proceedings of the Seventh International Conference on Music Information Retrieval, ISMIR 2006* pp. 216-221, Victoria, BC, Canada, USA, 2006.

[11] Peeling P., Li C., Godsill S., "Poisson point process modeling for polyphonic music transcription," *Journal of the Acoustical Society of America Express Letters* 121(4):EL168-EL175, April 2007.