M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 385

A Blackboard System for Automatic Transcription of Simple Polyphonic Music

Keith D. Martin
Room E15-401, The Media Laboratory
Massachusetts Institute of Technology
20 Ames St., Cambridge, MA 02139

Abstract

A novel computational system has been constructed which is capable of transcribing piano performances of four-voice Bach chorales written in the style of 18th century counterpoint. The system is based on the blackboard architecture, which combines top-down and bottom-up processing with a representation that is natural for the stated musical domain. Knowledge about auditory physiology, physical sound production, and musical practice has been successfully integrated in the current implementation. This report describes the system and its performance, highlighting its current limitations and describing some avenues of future work.

1 Introduction

Music transcription is a complicated cognitive task performed routinely by human musicians, but to date it has not been conquered by computer systems, except on toy problems. This paper describes a computational framework that may greatly expand the range of music that can be automatically transcribed by computer. In this introductory section, a brief noncomprehensive history of automatic transcription systems is presented, along with a high-level description of the type of blackboard architecture considered in this paper.

1.1 Transcription — definition and history

One functional definition of transcription is the act of listening to a piece of music and of writing down music notation for the notes that make up the piece. This abstraction ignores much of the nuance in music notation, but for purposes of this discussion, the parameters of concern are the pitches, onset times, and durations of the notes in a piece. These parameters are not sufficient to reproduce a perceptually equivalent "copy" of the original performance, as loudness and timbre are ignored (see [Scheirer 1995] for an attempt to achieve perceptual equivalence in score-guided transcriptions of piano performances), but they go a long way toward forming a useful symbolic representation of the music.

The history of automatic music transcription dates back at least 25 years. In the early 1970s, Moorer built a system for transcribing duets [Moorer 1975]. His system was limited, succeeding only on music with two instruments of different timbres and frequency ranges, and with strict limitations on the allowable simultaneous *intervals*[1] in the performance. Maher improved upon Moorer's system by relaxing the interval constraints, at the expense of requiring that the two instruments occupy mutually exclusive pitch ranges [Maher 1989, Maher 1990].

After Moorer, several systems were constructed which performed polyphonic transcription of percussive music, with varying degrees of success [Stautner 1982, Schloss 1985, Bilmes 1993]. In 1993, Hawley described a system which purported to transcribe polyphonic piano performances [Hawley 1993]. His approach was based on a differential spectrum analysis (similar to taking the difference of two adjacent FFT frames in a short-time Fourier transform) and was reported to be fairly successful, largely because piano notes do not modulate in pitch.

A research group at Osaka University in Japan has conducted research into automatic transcription for many years [Katayose and Inokuchi 1989]. Unfortu-

1. In this context, interval corresponds to the number of semitones separating two simultaneously sounded notes. Moorer's system was unable to detect octaves (intervals which are multiples of 12 semitones) or any other intervals in which the fundamental frequency of the higher note corresponds to the frequency of one of the overtones of the lower note.

nately, most of their published work is not yet available in English, and the available references do not describe their system in sufficient detail to allow a fair comparison with other existing systems. It appears that their system is capable of transcribing multiple voice music in some contexts.

1.2 Blackboard systems in brief

In parallel with transcription efforts, so-called "blackboard" systems were developed as a means to integrate various forms of knowledge for the purpose of solving ill-posed problems. The name "blackboard" comes from the metaphor of a number of experts standing around a physical blackboard, working together to solve a problem. The experts watch the solution evolve, and each individual expert makes additions or changes to the blackboard when his particular expertise is needed. In a computational blackboard system, there is a central workspace/dataspace called the blackboard, which is usually structured in an *abstraction hierarchy*, with "input" at the lowest level and a solution or interpretation at the highest. Continuing the metaphor, the system includes a collection of "knowledge sources" corresponding to the experts. An excellent introduction to the history of blackboard systems may be found in [Nii 1986].

1.3 A limited transcription domain

It is unrealistic to expect that an initial foray into polyphonic transcription will be successful on all possible musical input signals. Rather than attacking the broad problem of general transcription all at once, this paper presents an open-ended computational framework, which has currently been implemented to transcribe a small subset of tonal music, but which may be easily extended to deal with more complex musical domains.

The musical context chosen for this initial transcription system is piano performances of Bach's chorales. Bach wrote most of his chorales in four-voice polyphony (the standard bass-tenor-alto-soprano configuration). Today, they are used mostly to teach the principles of musical harmony to music theory students (in fact, they are often used as transcription exercises for such students), but they serve as an interesting and useful starting point because they embody a very structured domain of musical practice. The first phrase of an example Bach chorale is shown in Figure 1.

The importance of a structured domain is that it allows the transcribing agent to exploit the structure, thereby reducing the difficulty of the task. To give an example of such exploitation, consider a music theory student transcribing one of Bach's chorales. In a given chord, the student may find it quite easy to pick out the pitches of the bass and soprano notes by ear. It may be more difficult, however, to "hear out" the pitches of the two middle voices, even though it is quite easy to hear the *quality* of the chord. By taking advantage of knowledge of the chord quality, it is a simple task to determine which chord is being played, and it is possible to "fill in" the inner voices based on the surrounding context. The structure of 18th century counterpoint music provides powerful tools for transcription, which may be leveraged as easily by a computer knowledge-based system as by a human transcriber.

1.4 A sampling of relevant knowledge

The types of knowledge that may be usefully employed in the defined transcription task fall into three categories: knowledge about human auditory physiology, knowledge about the physics of sound production, and knowledge about the rules and heuristics governing tonal music in general and 18th century counterpoint in particular. Some knowledge examples are given below:

- From studies of human physiology, we know that the cochlea, or inner-ear, performs a running time-frequency analysis of sounds arriving at the eardrum.

- From physics, we know that the acoustic signals of pitched sounds, like musical notes, are very nearly periodic functions in time. Fourier's theory dictates that these sounds may be well-approximated by sums of sinusoids, which will show up as harmonic "tracks", or horizontal lines, in a time-frequency (spectrogram) analysis of the sound.

- From musical practice, we know many things about which notes may be sounded simultaneously. For example, we know that two notes separated by the musical interval of a diminished fifth (tri-tone) will not generally be sounded together.

The three examples given above are only a small portion of the available knowledge that may be exploited by a transcription system. The knowledge embodied in the current implementation will be presented in the next section.

2 Implementation

In this section, the implementation details of the transcription system are presented. The signal processing underlying the system's front end is described, followed by descriptions of the blackboard system control structure, data abstraction hierarchy and knowledge base.

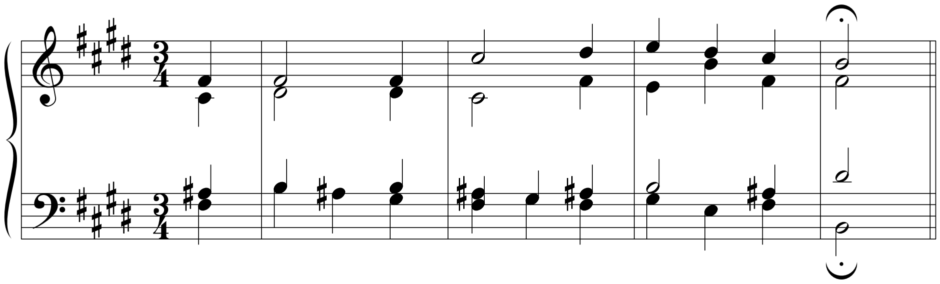

Figure 1: Music notation for the first phrase of a Bach chorale written in the style of 18th century counterpoint. The piece is titled *Erschienen ist der herrlich' Tag.*

2.1 The front end

The input to a real-world transcription system might be from a microphone or from the output of a recording playback device like a CD player. Equivalently, the system might analyze a stored computer soundfile. In the current system, a simple front-end has been constructed (in Matlab), which performs a time-frequency analysis of the sound signal and converts it into a simplified discrete representation. The blackboard system could easily be adapted to perform the front end processing, but it was much simpler, in this initial implementation, to rely upon the signal processing tools provided by Matlab.

The time-frequency analysis is obtained through the use of the short-time Fourier transform (STFT), which is equivalent to a filter-bank where the filter channels are linearly spaced in center frequency and all channels have the same bandwidth. This is a particularly gross model of the human cochlea (which is better modeled by a constant-Q filterbank), but is sufficient for the current application. The output of the STFT analysis of a piano performance of the Bach example is shown in Figure 2. The piano performance was synthesized from samples of a Bosendorfer grand piano by the CSound music synthesis language, based on a MIDI representation of the Bach score.

In parallel with the time-frequency analysis, the short-time running energy in the acoustic signal is measured by squaring and low-pass filtering of the signal. Sharp rises in the running energy are interpreted as note onsets and are used to *segment* the time-frequency analysis into chunks representing individual musical chords. Onset detection is a particularly fragile operation for real-world signals, but is sufficient for the computer-synthesized piano signals used in this system. In this initial implementation, simplicity is strongly favored over robustness in the

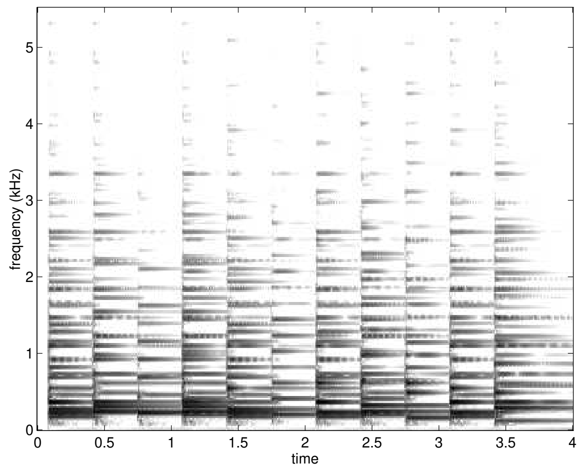

Figure 2: Short-term harmonic spectrum (spectrogram) for the musical example shown in Figure 1. The input to the blackboard system is a discretized version of the information in the spectrogram representation.

front end — rather, the concentration is placed on the blackboard portion of the system.

In each chord segment, the output of the time-frequency analysis is averaged over time, yielding an average *spectrum* for the chord played in that segment. Each segment spectrum is further summarized by picking the energy peaks in the spectrum, which correspond to the harmonic tracks (or stable sinusoids) in the sound signal. This particular style of front-end processing is useful mainly for piano performances of the type analyzed by this system (it takes advantage of the fact that piano notes do not modulate significantly in frequency), and is one of the most significant limitations in expanding the current system to handle other types of musical sounds.

The input to the blackboard system is a list of

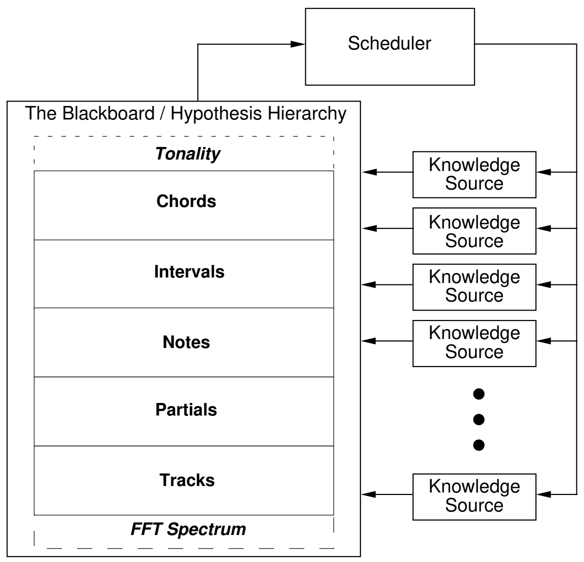

Figure 3: The control structure of the blackboard system described in this paper, also showing the data abstraction hierarchy.

"tracks" found by the front end. Each track has an associated onset time, frequency, and magnitude. The track data is stored in a text data file.

2.2 Blackboard control structure

As described in the Introduction, blackboard systems usually consist of a central dataspace called the blackboard, a set of so-called *knowledge sources* (KSs), and a scheduler. This is the implementation style that has been adopted for the current system. It is shown in outline in Figure 3.

Each knowledge source is made up of a precondition/action pair (much like the if/then structure of rule-based systems but procedural in nature). The knowledge sources are placed in a list, in decreasing order of expected benefit (the order is determined by the designer before compilation). The system operates in "steps": on each time step, the scheduler begins with the first knowledge source in the list and evaluates the precondition of each KS in turn. When a precondition is satisfied, the corresponding action is immediately performed and the sytem moves on to the next time step. If the KS list is ordered appropriately, only a small number of preconditions will be evaluated on average, and some unneeded computation is avoided. With the small number of knowledge sources and the relatively small number of hypotheses that are on the blackboard at a given time in the current implementation, this simple scheduler is fast enough to allow a user to watch the solution develop. As the system expands, it will become necessary to find more ways to increase efficiency (efficiency of computation in blackboard systems makes up an entire field of research and is far beyond the scope of this report).

It is worth noting that the simple scheduler used in this initial implementation ignores much of the power of the blackboard paradigm. Without modification, it would be difficult to introduce planning or other complex behaviors to the system.

2.3 Blackboard data abstraction hierarchy

In the current implementation, the blackboard workspace is arranged in a hierarchy of five levels. They are, in order of increasing abstraction: Tracks, Partials, Notes, Intervals, and Chords. At a lower level than Tracks, we can conceptually fit the raw audio signal and the spectrogram analysis performed by the front end, and at a higer level than Chords, we can conceive of more abstract musical data structures, like Chord Progressions or Tonality. These levels are not part of the current system, however. The abstraction hierarchy is drawn within the context of the blackboard control system in Figure 3.

In the current system, hypotheses are implemented in a frame-like manner. All hypotheses share a common set of slots and methods, including a list of supported hypotheses and a list of hypotheses that support the hypothesis object. Each hypothesis has a list of "Sources of Uncertainty", which will be described in a later section. They all have methods for returning the hypothesis's start time and rating.

2.3.1 Tracks

Track hypotheses, as described in earlier sections, are the raw data that the blackboard system analyzes. In addition to the common slots and methods, Track hypotheses have three slots, which are filled in from the data file generated by the front end: frequency, magnitude, and start time. The rating method returns a number between 0 and 1.0, depending on the value of the magnitude slot (a greater magnitude leads to greater rating — a heuristic metric).

2.3.2 Partials

Partial hypotheses bridge the gap between Tracks and Notes. They have one additional slot, the partial number. Several new methods are defined: idealFrequency returns the fundamental frequency of the supported Note multiplied by the partial number, and actualFrequency returns the frequency of the supporting Track. The rating function returns the rating of the supporting Track. By convention, a Partial may only support one Note and may only be supported by one Track.

In retrospect, it would seem that the Partial class is somewhat redundant. In a revised implementation, it might make more sense to add explicit "partial" slots to the Note hypotheses. Such a modification would simplify the implementation of at least one KS.

2.3.3 Notes

Note hypotheses have one additional slot: the pitch, given as the number of semitones distance from A5 (440 Hz). Additional methods include idealFrequency, which returns the ideal fundamental frequency of a note of the given pitch, based on the 440 Hz tuning standard, and actualFrequency, which returns an estimate of the fundamental frequency based on the frequencies of the supporting Partials.

The Note rating function is based on a simple evidence aggregation method, as described in [Davis *et al.*1977]. The first five Partials are considered, and each may contribute up to 0.4 evidence (either positive or negative). Both positive and negative evidence are tallied, and are then combined to form the rating. The amount contributed by each Partial is given by that Partial's rating multiplied by 0.4. This heuristic function fits many intuitions, but it fails to model high-pitched notes well. A better rating function might yield a significant improvement in the system's performance, as will be shown later.

2.3.4 Intervals

Interval hypotheses have three additional slots: the interval type, and the pitch classes of the two component notes. The rating function returns one of three values: 1.0, if there exist Note hypotheses that support both required pitch classes, 0.5 if only one pitch class is supported, and 0.0 if none are supported.

Interval hypotheses have a handful of auxilliary methods for determining whether given pitches "fit" into a given interval, and what the "canonical" interval is for two pitches (the only intervals of concern in the system right now are minor and major thirds, and perfect fifths, which are sufficient to construct the major and minor triads that are the primary building blocks of 18th century counterpoint music).

2.3.5 Chords

Chord hypotheses have four additional slots: the pitch of the chord's root and the three component Intervals. The rating function returns 1.0 if all three component Intervals have ratings of 1.0, and 0.0 otherwise. An auxilliary method is defined to take care of testing Intervals for possible chord membership.

2.4 Sources of uncertainty

As mentioned previously, each hypothesis maintains a list of "Sources of Uncertainty" (SOUs). They are in essence "tags" which are used either to direct the flow of the system's reasoning or for keeping some state information for the knowledge sources. When there are a large number of hypotheses on the blackboard, it can be quite computationally expensive to check the preconditions of all of the knowledge sources on every time step. By allowing the KSs to tag hypotheses with SOUs, the KSs may not have to re-evaluate their preconditions on every time step. Additionally, SOUs can be used to keep KSs from operating more than once on the same hypothesis, which can be problematic otherwise.

In some sense, the use of "sources of uncertainty" described above is an abuse of the term. In the IPUS system, from where the term is borrowed, the control system is based on the "Resolving Sources of Uncertainty" (or RESUN) architecture, which is a specific offshoot of the blackboard paradigm ([Dorken *et al.*1992, Klassner *et al.*1995, Winograd and Nawab 1995]). The current system does not use the RESUN architecture, though it could easily be modified to do so.

2.5 Blackboard knowledge base

The knowledge sources in the system fall under three broad areas of knowledge: garbage collection, knowledge from physics, and knowledge from musical practice (a fourth knowledge area, that of auditory physiology, is implicit in the front end). In this section, the thirteen knowledge sources which are present in the current system are briefly described in turn.

Figure 4 is a graphical representation of the knowledge base as a whole. It shows the hypothesis abstraction hierarchy used in the system with ten of the knowledge sources overlaid. Each KS is represented as a connected graph of nodes, where each node is a hypothesis on the blackboard, and the arrows represent something like a "caused a change in" relationship. The nodes where the arrows originate represent the hypotheses that satisfy the *precondition* of the KS; the nodes where the arrows terminate represent the hypotheses modified by the *action* of the KS. The white nodes represent competition hypotheses, which lie outside of the hypothesis abstraction hierarchy.

2.5.1 Track_NoExplanation

The `Track_NoExplanation` KS is the most primitive of the physics-based knowledge sources. It is used as

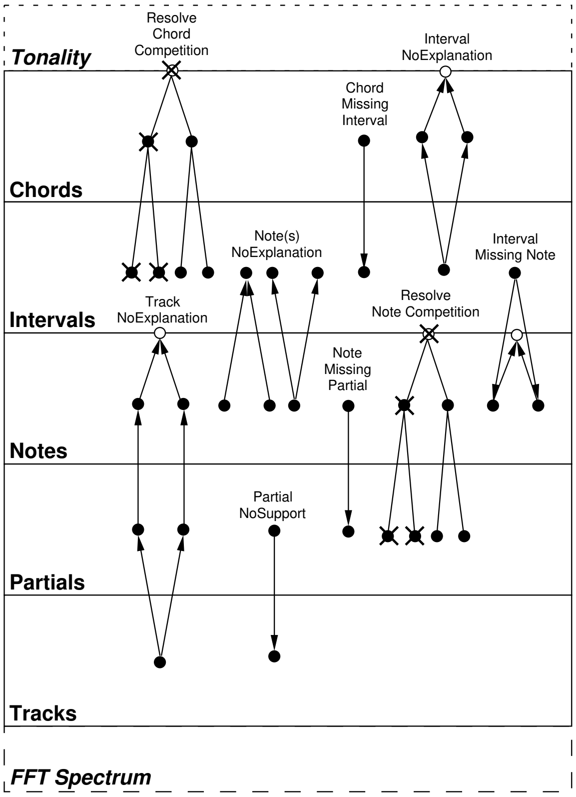

Figure 4: A graphical representation of the knowledge base as a whole. It shows the hypothesis abstraction hierarchy used in the system with ten of the knowledge sources overlaid. Each KS is represented as a connected graph of nodes, where each node is a hypothesis on the blackboard, and the arrows represent something like an "caused a change in" relationship. The nodes where the arrows originate represent the hypotheses that satisfy the *precondition* of the KS; the nodes where the arrows terminate represent the hypotheses modified by the *action* of the KS. The white nodes represent Competition hypotheses.

a last resort to create "bottom-up" pressure for the exploration of new note hypotheses when there is no "top-down" pressure.

Precondition: This KS searches through the list of active track hypotheses. If any are not attached to higher-level hypotheses, the precondition is satisfied by the one with the lowest frequency (from the TrackHyp *frequency* slot).

Action: The `Track_NoExplanation` KS creates a new CompetitionHyp and places it on the blackboard. It takes the TrackHyp's *frequency* slot and divides it by 1, 2, 3, 4, and 5. If any of the resulting frequencies are in the range of valid note pitches, NoteHyps are proposed at those pitches, and the TrackHyp is attached as supporting evidence, by way of newly created PartialHyps, as shown in Figure 4.

2.5.2 Note_MissingPartial

The `Note_MissingPartial` KS is another "physics-based" knowledge source. It creates top-down pressure to find support for a NoteHyp with empty partial slots.

Precondition: This KS searches through the list of active note hypotheses on the blackboard. The precondition is satisfied if it can find a NoteHyp with an empty *partial* slot (NoteHyps have implicit slots for their first ten partials).

Action: The `Note_MissingPartial` KS creates a new PartialHyp to support the selected NoteHyp, corresponding to the next partial in the note's harmonic series (up to an upper frequency limit of 2.5 kHz, a limit imposed by the system's front end).

2.5.3 Competition_SherlockHolmes

The name of the `Competition_SherlockHolmes` KS comes from Holmes's principle: if all other possibilities have been eliminated, the remaining one must be correct. This KS performs a form of "garbage collection".

Precondition: This KS searches the blackboard for active competition hypotheses. The precondition is satisfied if it can find one that has only one active supporting hypothesis.

Action: The `Competition_SherlockHolmes` KS removes the selected CompetitionHyp from the blackboard.

2.5.4 Partial_NoSupport

The `Partial_NoSupport` KS is another "physics-based" knowledge source. It creates top-down pressure to find support for newly created partial hypotheses.

Precondition: The precondition is satisfied if there is a partial hypothesis on the blackboard with no supporting track hypothesis.

Action: This KS's action is to search through the track hypotheses on the blackboard for the one that matches the expected frequency of the selected PartialHyp most closely. If it can not find one within 30 Hz (an arbitrary threshold), it sets the PartialHyp's rating to (-1.0), indicating that no match was found. If a match is found, then the TrackHyp is attached to the PartialHyp as support.

2.5.5 Note_OctaveError

The `Note_OctaveError` KS embodies a "physics-based" piece of knowledge. If a Note hypothesis has much stronger even-numbered partials than odd-numbered partials (measured by an empirical threshold), then it is likely that a Note one octave higher in pitch is a better match to the data.

Precondition: This KS searches through the active note hypotheses on the blackboard. It examines all of the note hypotheses that have all of their partial slots filled. In each case, it averages the magnitudes of the first three odd numbered partials and the first three even numbered partials. If the even numbered partials' average magnitude is more than 6 dB greater than that of the odd numbered partials, then the precondition is satisfied.

Action: This KS's action is to remove the selected NoteHyp from the blackboard, and to create a new note hypothesis, with a pitch one octave higher, placing it on the blackboard, if there is not already an note with that pitch on the blackboard.

2.5.6 Note_PoorSupport

The `Note_PoorSupport` KS performs a form of garbage collection. It removes invalid note hypotheses from the blackboard. Its precondition is tested after that of the `Note_OctaveError` KS, so that octave errors are detected before Notes are discarded.

Precondition: This KS's precondition is satisfied if there is a NoteHyp on the blackboard, whose partial slots are all filled, but whose rating is below a cutoff threshold (empirically set to 0.6).

Action: The selected NoteHyp is removed from the blackboard, along with its supporting PartialHyps.

2.5.7 Notes_NoExplanation

The `Notes_NoExplanation` KS embodies a piece of musical knowledge: any two notes played simultaneously form a musical interval, defined by the difference between their pitches mapped onto a discrete set of interval types.

Precondition: If there are two simultaneously occurring NoteHyps on the blackboard, both of which having all of their partial slots filled, and neither supporting a higher-level hypothesis, then the precondition is satisfied.

Action: This KS's action is to place a new IntervalHyp on the blackboard, and to attach the two selected NoteHyps as supporting evidence.

2.5.8 Note_NoExplanation

The `Note_NoExplanation` KS is a companion to the `Notes_NoExplanation` KS. It performs its action when there is already an IntervalHyp on the blackboard, and there is a single unexplained NoteHyp.

Precondition: The precondition is satisfied if there is an active NoteHyp on the blackboard, with all of its Partial slots filled, and there is a simultaneous IntervalHyp on the blackboard.

Action: This KS's action is somewhat complicated. First, the NoteHyp is tested against all existing IntervalHyps. It is attached to any that it fits into. If it does not fit into any, then the NoteHyp represents a new pitch class, and a set of new IntervalHyps is created with all of the distinct simultaneously occurring pitch classes.

2.5.9 ResolveNoteCompetition

The `ResolveNoteCompetition` KS embodies a form of garbage collection. It disables competitions between notes when one of the competing notes is found to be "valid". The other competing notes are not removed from the blackboard, but they are not actively investigated unless they are reactivated by another knowledge source.

Precondition: The precondition is satisfied if there is an active competition between NoteHyps on the blackboard and one of the competing NoteHyps has all of its partial slots filled and a rating above the note acceptance cutoff.

Action: The selected CompetitionHyp is taken off the blackboard. The selected NoteHyp is accepted as a confirmed hypothesis, and the remaining NoteHyps are deactivated, but not removed from the blackboard.

2.5.10 Interval_NoExplanation

The `Interval_NoExplanation` KS embodies a form of musical knowledge. It creates new chord hypotheses when an IntervalHyp does not fit into existing ChordHyps.

Precondition: The precondition is satisfied if there is an active IntervalHyp on the blackboard that does not support any higher-level hypotheses.

Action: This KS examines the selected IntervalHyp and determines which triads it could be a member of (intervals of a minor third, major third, or perfect fifth can all be component parts of both major and minor triads). The KS places a new CompetitionHyp on the blackboard, connected to two new ChordHyps, both of which are supported by the selected IntervalHyp.

2.5.11 Chord_MissingInterval

The `Chord_MissingInterval` KS embodies musical knowledge. It creates top-down pressure to find support for a chord hypothesis.

Precondition: The precondition is satisfied if there is an active ChordHyp on the blackboard that is missing one or more of its three component intervals (m3, M3, P5).

Action: This KS places the missing interval(s) on the blackboard (as predictions).

2.5.12 ResolveChordCompetition

The `ResolveChordCompetition` KS performs a garbage collection task. It removes unnecessary chord competitions from the blackboard.

Precondition: The precondition is satisfied if there is an active competition between multiple chords on the blackboard, and one of the chords has a rating of 1.0.

Action: The KS removes the selected Competition-Hyp from the blackboard, and deactivates the supporting ChordHyps with ratings below 1.0. Deactivated ChordHyps are not removed from the blackboard.

2.5.13 Interval_MissingNote

The `Interval_MissingNote` KS embodies a piece of musical knowledge. It creates top-down pressure to find support for interval hypotheses.

Precondition: The precondition is satisified if there is an active IntervalHyp on the blackboard that is missing one or more of its component notes.

Action: The KS searches through the active NoteHyps on the blackboard to fill in the note slots in the selected IntervalHyp. If any note slots are not filled in, the KS creates new NoteHyps for the missing pitch class and places them in competition.

3 An example transcription

In this section, the system's transcription of the Bach example from Figures 1 and 2 is presented in the form of a brief annotation of its progression from start to finish.

3.1 Steps 1–63 - Finding the First Note

When the system starts, there are no hypotheses on the blackboard, so it reads a block of track data (corresponding to the first chord) from the front end output and places the data on the blackboard in the form of Track hypotheses (TrackHyps). Thus, at the first time step, there are 25 TrackHyps on the blackboard, all with "No Explanation" SOUs (meaning that they do not support any higher level hypotheses).

On the first time step, the precondition for the `Track_NoExplanation` knowledge source (KS) is satisfied and it performs its action, which in general places several competing note hypotheses (NoteHyps) on the blackboard. In this case, the action adds only one NoteHyp to the blackboard because the frequency of the TrackHyp is such that it can only support one note in the range which the system considers (basically, notes that fall on the staff). As an artifact of the programming style, the NoteHyp is still attached to a Competition hypothesis (CompetitionHyp), even though there are no other NoteHyps in competition. The unnecessary CompetitionHyp is removed from the blackboard on the second time step. A screen shot of the system after the first step is shown in Figure 5.

In the next set of time steps, the `Note_MissingPartial` and `Partial_NoSupport` KSs are activated alternately, as the blackboard seeks to find evidence to support the note hypothesis proposed in Step 1. In this case, the first partial of the bass note was sufficiently sharp (a common characteristic of string timbres) to confuse the system into searching for the wrong note (G instead of $F\sharp$). Because the G note is not actually present in the signal, several Partials do not have any support, as can be seen in Figure 6, which shows the state of the system at Step 21. On Step 22, NoteHyp1 is removed from the blackboard due to its low rating, and the search for notes begins again on Step 23 with the expansion of TrackHyp2, which the system hypothesizes is the first partial of NoteHyp2 ($A\sharp4$) or the second partial of NoteHyp3 ($A\sharp3$).

In steps 24–42, the supporting PartialHyps for NoteHyp3 are filled in. TrackHyps were not found to support two of the PartialHyps, so the NoteHyp's rating is fairly low. On step 43, the precondition for the `Note_OctaveError` KS is satisfied, and NoteHyp3 is removed from the blackboard.

NoteHyp2 is explored during steps 44–63. After all of its PartialHyps are filled in, NoteHyp2's rating is 0.712, which is above the cutoff threshold; NoteHyp2 is therefore accepted as a confirmed hypothesis. The note's color is changed from grey to black in the output display, as shown in Figure 7.

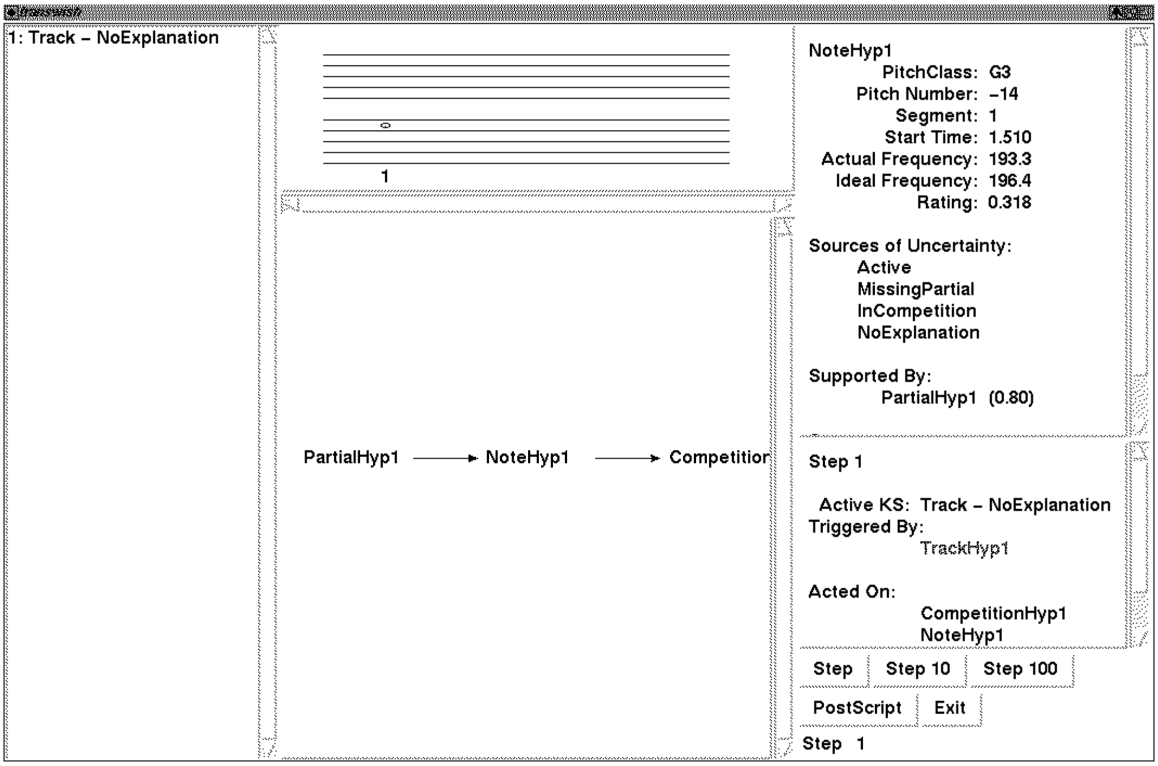

Figure 5: A screen capture of the blackboard system at Step 1. The panel on the left hand side contains a history of the knowledge sources that have executed at each blackboard time step. In the center, the graphical output of the system sits atop a graphical representation of a part of the blackboard data hierarchy (the arrows are used to indicate "supports" relationships). On the right hand side, there are two panels: the top one contains a detailed description of a selected hypothesis; the bottom one contains a summary of the actions performed on the most recent time step. At the bottom of the right hand side are buttons used to control the operation of the system. In Step 1, a Note hypothesis (NoteHyp1) was proposed to explain a Track hypothesis (TrackHyp1).

3.2 Steps 64–101 - A Second Note Leads to an Interval

During steps 64–84, NoteHyp5 ($C\sharp 3$) is explored and discarded as invalid. During steps 85–100, NoteHyp4 is explored and accepted as valid. In step 101, NoteHyp2 and NoteHyp4, the first two confirmed notes, are joined together into an Interval hypothesis (IntervalHyp1), as shown in Figure 8, by the `Notes_NoExplanation` KS, whose precondition is satisfied when there are two simultaneous notes on the blackboard which do not support any higher level hypotheses. The two notes form an interval of a minor third.

3.3 Steps 102–180 - Determining the First Chord

On step 102, the new interval hypothesis satisfies the precondition of the `Interval_NoExplanation` KS, which proposes two competing chord hypotheses (ChordHyp1 [$F\sharp$ Major] and ChordHyp2 [$A\sharp$ minor]) to account for the interval hypothesis.

On step 103, the precondition for the `Chord_MissingInterval` KS is satisfied by both chord hypotheses. In this case, the KS acts upon ChordHyp2, placing IntervalHyp2 and IntervalHyp3 on the blackboard, corresponding to the missing major third and perfect fifth of the $A\#$ minor triad.

At step 104, IntervalHyp3 satisfies the precondition of the `Interval_MissingNote` KS, and three NoteHyps are posted on the blackboard, corresponding to the F pitch needed to complete the Interval. In steps 105–153, the system explores NoteHyps 6, 7, and 8. None are valid and they are removed from the blackboard.

On step 154, the `Chord_MissingInterval` KS is activated by ChordHyp1, resulting in the posting of IntervalHyp4 and IntervalHyp5 on the blackboard. On step 155, the `Interval_MissingNote` KS is activated by IntervalHyp5, resulting in the posting of NoteHyp9,

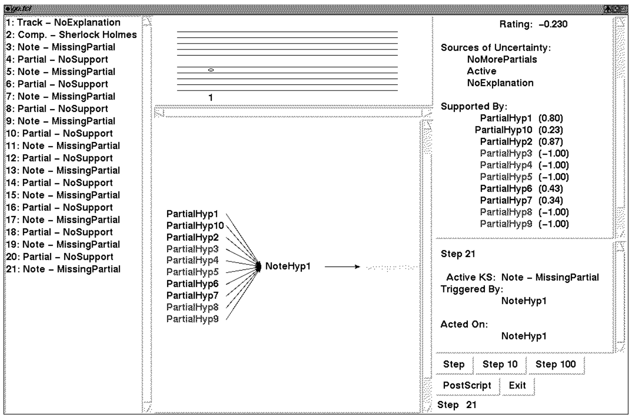

Figure 6: A screen capture of the blackboard system at Step 21.

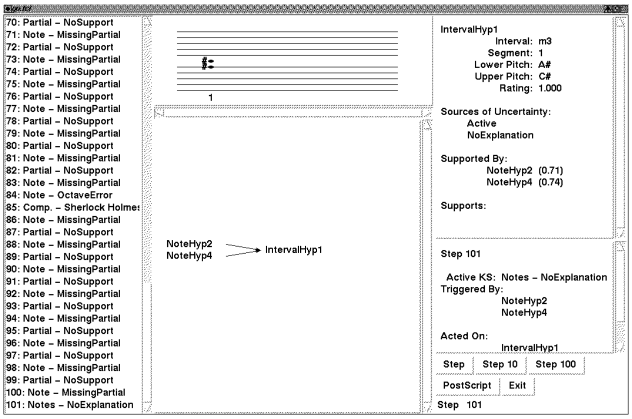

Figure 8: A screen capture of the blackboard system at Step 101.

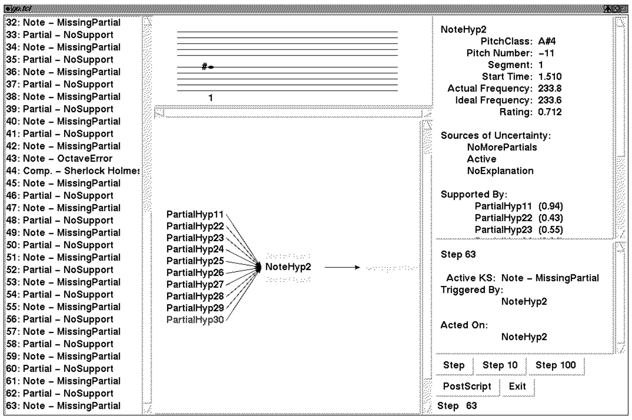

Figure 7: A screen capture of the blackboard system at Step 63.

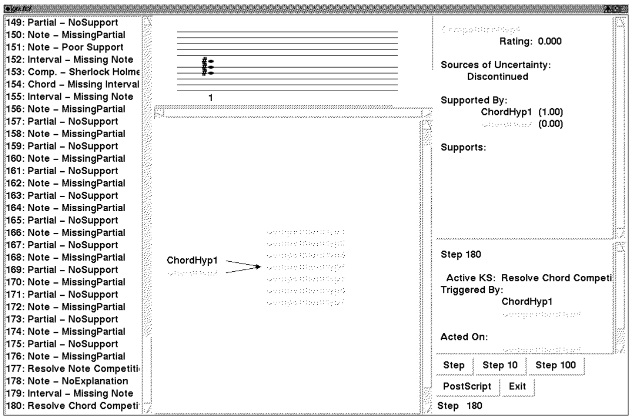

Figure 9: A screen capture of the blackboard system at Step 180. ChordHyp1 is accepted as a correct hypothesis.

NoteHyp10, and NoteHyp11 on the blackboard, corresponding to the three $F\#$s in the acceptable pitch range.

In steps 156–176, NoteHyp11 is explored. It is accepted as valid, added as support for IntervalHyps 4 and 5, leading to the acceptance of ChordHyp1 on step 180, as shown in Figure 9.

3.4 Step 181 - One Note missed ...

On step 180, none of the KS preconditions are satisfied, so the TrackHyp data for the second chord segment is loaded from the input file and placed on the blackboard. The preconditions are re-tested, and the action of the `Track_NoExplanation` KS is fired.

The system has made an error, however, which reveals its primary weakness (not coincidentally, the primary weakness of all polyphonic transcription systems to date). As is apparent by looking back at Figure 1, the system has missed the $F\#$ in the soprano voice, one octave above the $F\#$ in the bass voice. The system, as currently formulated, will not detect the higher note in any octave relation. This effect is due to the physics of sound production (in that the upper note in the octave relation shares all of its partials with the lower note), and the system will require more musical knowledge or a better note "model" in order to overcome it.

3.5 Steps 182–1712 – Business as usual

In steps 182–1711, the system progresses through the first seven segments of the performance. In the fifth segment, a $C\#$ in the soprano voice is missed because it is one octave above the alto voice.

Toward the end of the sequence of actions, NoteHyp99 is explored. The state of the system after step

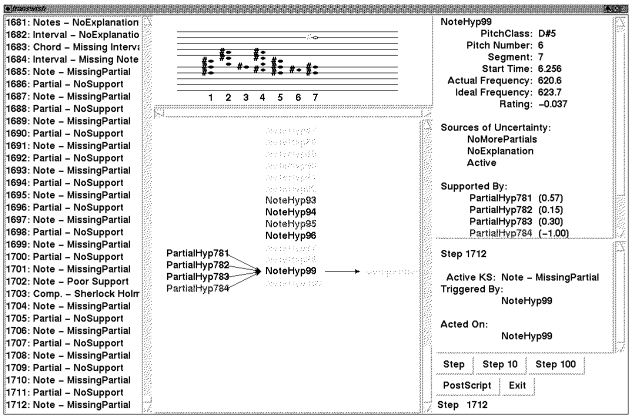

Figure 10: A screen capture of the blackboard system at Step 1712.

1712 is shown in Figure 10.

3.6 Step 1713 – Another failure

On step 1713, NoteHyp99 is removed from the blackboard by the `Note_PoorSupport` KS, due to a relatively low rating. This, however, is an error, since the $D\#5$ pitch represented by NoteHyp99 is actually one of the notes played in the performance.

The failure to accept NoteHyp99 is due to a modeling error in the Note hypothesis data class. The NoteHyp rating function is a heuristic function of the ratings of the supporting PartialHyps. The same function is used for NoteHyps of all pitches — herein lies the problem. It turns out that the higher notes on a piano keyboard tend to have very weak upper partials, a fact that is not taken into account by the current rating function.

3.7 And so on ...

The rest of the example proceeds as expected. Three additional high-pitched notes are incorrectly removed from the blackboard, and two more octave mistakes are made. The resulting transcription is presented in two forms. The first, a textual representation, is a simple list of detected notes with their pitches and onset times, as shown in Figure 11. A graphical display of the note onset data, presented in a manner that is more amenable to comparison with the original musical score, is shown in Figure 12.

4 Conclusions

In this section, the limitations of the current system will be described, followed by a description of the system's successes, an assessment of the progress that has been made toward a useful system for transcribing real-world musical signals, and a brief outline of some of the next few goals in the current line of research.

As described in the Introduction, the original goal in designing the present system was to build a system capable of transcribing piano music in the style of 18th century counterpoint. The system, as it is implemented today, solves a slightly different problem than was originally proposed: it can transcribe synthesized piano performances in which no two notes are ever played simultaneously an octave apart and where all notes are within the somewhat limited range of $B3$ (123 Hz) to $A5$ (440 Hz).

NoteHyp2	A#4	1.510
NoteHyp4	C#4	1.510
NoteHyp11	F#3	1.510
NoteHyp13	B4	2.317
NoteHyp17	D#4	2.317
NoteHyp24	F#4	2.317
NoteHyp34	A#4	3.111
NoteHyp38	B4	3.870
NoteHyp40	D#4	3.870
NoteHyp43	G#3	3.870
NoteHyp45	F#4	3.870
NoteHyp54	C#4	4.665
NoteHyp61	F#3	4.665
NoteHyp67	A#4	4.665
NoteHyp77	G#3	5.501
NoteHyp94	F#3	6.256
NoteHyp96	A#4	6.256
NoteHyp104	B4	7.052
NoteHyp106	E4	7.052
NoteHyp114	G#3	7.052
NoteHyp130	E3	7.855
NoteHyp139	F#3	8.627
NoteHyp141	A#4	8.627
NoteHyp152	B3	9.411

Figure 11: Text output of the blackboard system for the Bach example.

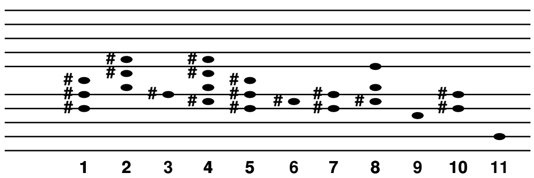

Figure 12: Graphical output of the blackboard system for the Bach example.

This change of scope is a result of two problems. First, the failure to correctly detect octaves (as demonstrated in the annotated example) is due to physical ambiguity and to a lack of musical knowledge. It might be possible to correct this deficiency with one or two new knowledge sources. In 18th century counterpoint, there should be four notes in every chord. If only three are detected directly, then the fourth will generally be playing in unison with, or an octave above, one of the other notes.

The second problem is due to a poor assumption in the rating function for note hypotheses. It turns out that the higher notes in a piano's range do not have strong upper partials (and some of the partials may well be above the 2.5 kHz cut-off imposed by the current front end implementation). The note hypothesis rating function, however, does not take this into account, and therefore note hypotheses with pitches above $A5$ (440 Hz) will not generally have a high-enough rating to be accepted as valid. This problem might be fixed by a careful reimplementation of the rating function.

A final limitation of the current system is imposed by the front end. It tacitly makes several assumptions about the acoustic signal, namely that all notes in a chord are struck simultaneously and that the sounded notes do not modulate in pitch. These limitations might be addressed by a more complicated initial signal analysis, perhaps like the track analysis described by Ellis [Ellis 1992].

While the current system suffers from a number of limitations, it marks an important first step toward a working automatic transcription tool. The flexibility of the blackboard approach is its greatest asset, making it possible to seamlessly integrate knowledge from multiple domains into a single system. The result is open-ended, allowing for great ease in implementing future extensions.

In its current form, the blackboard transcription system is capable of analyzing piano performances with multiple simultaneously sounded notes, with the limitations just described. It begins with a "jumbled grab-bag" of partials and successfully detangles them, identifying the component notes and the chords that they make up. In order to improve the current system to a point at which it will be useful as an automated transcription tool for real-world musical signals, a number of extensions must be made. The current level of musical knowledge in the system is minimal, so one obvious direction is to extend it further by including knowledge about tonality (which would reduce the number of unneccessarily explored note hypotheses) and about melodic motion (which might further reduce the computational load by helping the system make better predictions).

Currently, we are rethinking the computational approach taken in this research, with the explicit goal of constructing a system that more closely resembles human musical understanding. Such a system will "perceive" chord quality and tonality more directly, perhaps through a mechanism based on the correlogram [Slaney and Lyon 1993]. Weft analysis also appears to be a worthy area of pursuit [Ellis 1995].

References

[Bilmes 1993] Jeff Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, MIT Media Laboratory, 1993.

[Davis *et al.*1977] Randall Davis, Bruce Buchanan, and Edward Shortliffe. "Production Rules as a Representation for a Knowledge-Based Consultation Program". *Artificial Intelligence*, 8:15–45, 1977.

[Dorken *et al.*1992] Erkan Dorken, Evangelos Milios, and S. Hamid Nawab. "Knowledge-Based Signal Processing Application". In Alan V. Oppenheim and S. Hamid Nawab, editors, *Symbolic and Knowledge-Based Signal Processing*, chapter 9, pages 303–330. Prentice Hall, Englewood Cliffs, NJ, 1992.

[Ellis 1992] Daniel P. W. Ellis. A perceptual representation of audio. Master's thesis, Massachusetts Institute of Technology, February 1992.

[Ellis 1995] Daniel P. W. Ellis. Mid-level representation for computational auditory scene analysis. In *Proc. of the Computational Auditory Scene Analysis Workshop; 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

[Hawley 1993] Michael Hawley. *Structure out of Sound*. PhD thesis, MIT Media Laboratory, 1993.

[Katayose and Inokuchi 1989] Haruhiro Katayose and Seiji Inokuchi. "The Kansei Music System". *Computer Music Journal*, 13(4):72–77, 1989.

[Klassner *et al.*1995] Frank Klassner, Victor Lesser, and Hamid Nawab. "The IPUS Blackboard Architecture as a Framework for Computational Auditory Scene Analysis". In *Proc. of the Computational*

Auditory Scene Analysis Workshop; 1995 International Joint Conference on Artificial Intelligence, Montreal, Quebec, 1995.

[Maher 1989] Robert Crawford Maher. *An Approach for the Separation of Voices in Composite Musical Signals*. PhD thesis, University of Illinois at Urbana-Champaign, 1989.

[Maher 1990] Robert C. Maher. "Evaluation of a Method for Separating Digitized Duet Signals". *J. Audio Eng. Soc.*, 38(12), December 1990.

[Moorer 1975] James A. Moorer. *On the segmentation and analysis of continuous musical sound by digital computer*. PhD thesis, Department of Music, Stanford University, Stanford, CA, May 1975.

[Nii 1986] H. Penni Nii. "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures". *The AI Magazine*, pages 38–53, Summer 1986.

[Scheirer 1995] Eric D. Scheirer. Extracting expressive performance information from recorded music. Master's thesis, Program in Media Arts and Science, Massachusetts Institute of Technology, 1995.

[Schloss 1985] W. Andrew Schloss. *On the Automatic Transcription of Percussive Music — from Acoustical Signal to High-Level Analysis*. PhD thesis, CCRMA - Stanford University, May 1985.

[Slaney and Lyon 1993] Malcolm Slaney and Richard F. Lyon. "On the importance of time — a temporal representation of sound". In Martin Cooke, Steve Beet, and Malcolm Crawford, editors, *Visual Representations of Speech Signals*, pages 95–116. John Wiley & Sons, 1993.

[Slaney 1995] M. Slaney. "A critique of pure audition". In *Proc. of the Computational Auditory Scene Analysis Workshop; 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

[Stautner 1982] John Stautner. The auditory transform. Master's thesis, MIT, 1982.

[Winograd and Nawab 1995] Joseph M. Winograd and S. Hamid Nawab. "A C++ Software Environment for the Development of Embedded Signal Processing Systems". In *Proceedings of the IEEE ICASSP-95*, Detroit, MI, May 1995.

[Winograd 1994] Joseph M. Winograd. Ipus c++ platform version 0.1 user's manual. Technical report, Dept. of Electrical, Computer, and Systems Engineering, Boston University, 1994.