Transcriber: A System for Automatically Transcribing Musical Duets

Cory McKay
Faculty of Music, McGill University
555 Sherbrooke Street West
Montreal, Quebec, Canada H3A 1E3

Wes Hatch
Faculty of Music, McGill University
555 Sherbrooke Street West
Montreal, Quebec, Canada H3A 1E3

Abstract

A system is presented that automatically transcribes synthesized clarinet and oboe audio data. A front-end was used that made use of sinusoidal tracks as well as a gammatone filterbank. Arrays of neural networks were then used to process the data produced by the front-end into a score. The system performed very well in correctly identifying pitches and in avoiding missed notes and false notes. The rhythm of the transcriptions was sometimes irregular, however. Problems that have been encountered in other transcription systems, such as octave misidentification, difficulties with notes of short duration and dependence on stylistic qualities were avoided. The potential for scaling the system to more difficult tasks given more time and computing resources is discussed. A variety of techniques were explored and are discussed here.

1. Introduction

Automated general-purpose polyphonic music transcription systems have a great deal of potential utility, both to music technologists and to traditional music researchers who would find it convenient to avoid having to manually transcribe performances. Although there has been some success with monophonic transcription, a widely-accepted polyphonic system has yet to be implemented. There is thus ample opportunity for researchers to build on the work that has already been done in this field.

Ideally, a polyphonic transcription system would take in an arbitrary musical audio signal and produce a notated score, complete with pitch, rhythm, dynamics and tempo information for each voice in the signal. Such a system could perhaps even glean information from audio signals that is not traditionally notated, but is perceived by humans.

Unfortunately the difficulties related to extracting precise and reliable data from audio data make this an unrealistic goal, at least for the moment. Problems such as spectral variations within a given instrument, difficulties in identifying short notes due to the prevalence of transients, voice crossing and difficulties in identifying the correct octave of a note have thus far prevented the successful implementation of a general-purpose transcription system.

It was therefore decided to attempt to solve a simplified version of this problem in order achieve intermediate results that could give insights into techniques that could be applied to the general problem of blackboard transcription. The first simplification was to limit the number of voices to one or two. A solution to this problem is likely simpler than transcribing arbitrary numbers of instruments, while at the same time still necessitating a solution that deals with the same types of problems that are encountered with general transcription. This limitation therefore makes it possible to explore the relative effectiveness of different techniques, which in turn could give insights into which approaches should be studied at further length in order to be applied to the problem of general transcription.

The second simplification was to specify which instruments would be present. This not an impractical limitation, given that identification of instruments can be a difficult task for computers but is relatively simple for trained humans. A user could simply specify different modules to transcribe different combinations of instruments. A semi-automatic system that requires the user to specify which instruments are present would greatly simplify the task of the computer while at the same time requiring only a moderate contribution from the human user.

The third simplification was to use synthesized audio rather than real audio recordings. This is the most troublesome simplification, as synthesized audio is much more consistent than real-world audio and it is therefore much easier to successfully apply pattern matching techniques to it. Nonetheless, this approach is consistent with the goal of simplifying the problem in order to study the effectiveness of different techniques.

One final note is that the transcribed output contains only rhythm and pitch information. All other information, including dynamics, is ignored. This is not considered to be a serious limitation, at least at this stage, since pitch and rhythm are arguably the types of information best presented by traditional scores.

One approach that has been used elsewhere that was avoided here was the incorporation of high-level musical knowledge into the system. This includes information such as common chord transitions or melodic devices. Although this kind of knowledge is both useful and appropriate for transcription systems, it inherently limits the system to specific styles of music or particular musical traditions. It was therefore decided to avoid this approach since it was our goal to develop a system that could eventually be adapted to transcription in general. Of course, it would always be possible to use a music theory system to process and improve the results of our system if a particular user were interested only in one type of music.

This paper will first present a review of background material and research in the area of polyphonic transcription. The implementation of the system will then be discussed and the different architectures and techniques that were used will be presented. The experimental results will then be given and discussed. This paper will then be ended with some final conclusions and a discussion of how this research can be expanded in the future.

2 Background Research

It seems that much progress has been made on the problem of polyphonic transcription since early attempts in the mid-70's. However, the success of almost all systems since then has been as a result of at least one significant simplification. As will be seen, systems have been implemented that transcribe audio signals derived from only one or a few instruments, that extract only a given instrument from a complex signal or that deal with music that obeys very restrictive rules. In addition to these limitations, most systems that have been implemented produce very simplified scores, often with information limited to pitch and note onset time for each voice. Such systems have reportedly achieved success rates of between 70% and 90%, although these numbers may be inflated due to the limited testing suites that have been applied.

Although there have been a wide variety of methodologies applied to polyphonic transcription, certain techniques have gained prominence in recent years. With respect to front-end processing, two general approaches have come to the fore. Sinusoidal-based analysis and correlation-based techniques have laid the groundwork and show promise for future implementations. This is doubly important when we consider the limitation that a poor representation of data places on the rest of the system—without reliable data from the front-end, the rest of the system's functionality is extremely limited.

The front-end analyzes PCM audio data and reinterprets it so that it becomes meaningful to other components within the transcription system. Perhaps the most promising implementation towards this goal comes via Fourier-based analysis techniques.

Bello *et al.* (2000) describe a system that receives the averaged STFT of a signal and identifies the peaks in the spectrum. These peaks are stored as tracks; they contain frequency and magnitude information that the system follows over time. These sinusoidal tracks, as they're referred to, provide a good indication of how the frequency content of a given sound is evolving over time.

The advantage of this method over a traditional STFT is that there is no loss of resolution towards the lower frequency ranges. A Fourier transform provides information in bins evenly spaced in frequency, the result of which is that more information is provided in the higher frequency ranges, yet there remains poor resolution and uncertainty in the low frequencies. Contrariwise, utilizing sinusoidal tracks allows only the most salient information to be identified and extracted from a signal, wherever it may lie in the frequency spectrum.

Dixon (2000) uses a similar technique to circumvent this problem. A tracking phase vocoder is employed in order to alleviate some of the uncertainty in the low notes. Rather than using the centre frequency of each bin, a more accurate estimate is attained by using phase information obtained from adjacent FFT windows. Specifically, in the bins surrounding a spectral peak, the rate of phase change will correspond to the actual frequency present. As before, the peaks in the spectrum and their frequency are then combined to give, as Dixon describes it, "atoms of energy localized in time and frequency." However, because of the similarity of the output to sinusoidal tracks, this representation may be viewed as an alternative to sinusoidal-track implementation.

We have described a rather elaborate method to extract information from the audio signal while at the same time preserving low frequency resolution. Of course, a longer analysis window would certainly help quantify frequencies in the lower frequency range; however, this comes at the expense of creating another problem whereby there is insufficient resolution in time. Thus, forming sinusoidal tracks allows salient information to be presented while at the same time maintaining resolution in time.

An altogether different structure for a front-end employs correlation-based analysis. In general, these methods seek to model human perception by filtering the audio data in a similar way as the ear. A gammatone filterbank usually provides a basis for this type of analysis (Martin, 1996b; Martolt, 2001). First, the audio signal is decomposed into a number of frequency bands with near-constant-Q bandwidth in the middle and high frequencies. The frequency and width of each band closely resembles equivalent bands on the basilar membrane. Next, the output of each band is usually processed by a model of the inner hair cells of the cochlea. This process may be viewed as "a half-wave rectification followed by smoothing … and onset enhancement" (Martin, 1996b). Often, further analysis is performed by short-time auto-correlation. One variation includes using a bank of filters to produce log-lag correlograms, and then determining pitch by measuring the periodic energy in each filter channel as a function of lag (Martin, 1996b). Martin claims that this approach makes the bottom-up detection of octaves possible. This could potentially serve as an improvement over autocorrelation, as it more closely relates the variation of human pitch resolution ability. However, while this technique was implemented by Martin, he did not achieve any definitive experimental results indicating whether this approach is actually an improvement.

One advantage of using a front-end that models human perception in this way is that the many nuances of human perception may be accounted for. These include the "missing fundamental," and weak pitch perception resulting from interrupted noise bursts. However, critics of this approach (Klapuri, 1998) cite difficulties arising from auditory stream analysis. Creating a model based on the human ear means that it is subjected to the same, inherent problems of perception that humans are faced with. For instance, with autocorrelation a fusion of information on perceptual grounds (the sound of several instruments combining to form the perception of a single timbre) restricts the separate treatment of each harmonic partial. Klapuri claims on this basis that transcription systems equipped with a correlation-based front-end would, in theory, suffer from this, too.

The SONIC system (Martolt, 2001) attempts to extend the effectiveness of a correlation-based front-end. It first uses a gammatone filterbank and a hair model to produce an output on many frequency channels. Instead of using autocorrelation or some kind of peak-picking algorithm next, a network of adaptive oscillators is employed. These oscillators adapt their phase and frequency in response to an input; when a periodic signal is sent as input, it tries to model that input by adjusting its phase and frequency. Partial tracks may then be formed by observing the output of each oscillator. This results in a very robust front-end, leading to perhaps one of the most successful transcription systems to date.

A number of methods have been used to derive transcriptions from the output of front-end systems. One of the most common approaches is to use a blackboard systems. The term "blackboard" comes from the notion of a group of experts standing around a blackboard working together to solve a problem. Each expert writes contributions on the blackboard based on his/her expertise. The experts watch the problem evolve on the blackboard until a solution is achieved. In terms of computing, the "blackboard" is a central dataspace that is usually arranged in a hierarchy so that input is at the lowest level and output is at the highest. The "experts" are called "knowledge sources," and they generally consist of a set of heuristics and pre-conditions whose satisfaction results in a hypothesis that is written to the blackboard. Each knowledge source forms hypotheses based on information from the front-end of the system and hypotheses presented by other knowledge sources. The problem is considered solved when all knowledge sources are satisfied with all hypotheses on the blackboard to within a given margin of error.

Keith Martin was one of the first researchers to apply blackboard systems to music transcriptions (Martin, 1996 a). His system was limited to analyzing performances of piano music and was tested on four-voice Bach chorales. The front-end of Martin's system applied short-time Fourier transforms to the input signal to generate associated sets of onset times, frequencies and amplitudes that were fed to the blackboard system. The blackboard system consisted of thirteen knowledge sources, each falling into one of three types: garbage collection, physics and musical practice. The hypotheses made by the knowledge sources fell into five hierachally-organized classes, namely tracks, partials, notes, intervals and chords. A sequential scheduler was used to coordinate the knowledge sources. One of the greatest weaknesses of this system was that it tended to misidentify octaves. In order to resolve this problem, Martin proposed using a bank of filters to produce log-lag correlograms, as described above (1996b). The correlograms could then be fed as the basic unit to the blackboard system. Martin did not achieve any definitive experimental results indicating whether this approach is better than his original approach.

Bello and Sandler (2000) have designed a system based on Martin's design, using a sequential scheduler. Aside from refining the knowledge sources and adding high-level musical knowledge, they implemented a chord recognizer knowledge source as a feed-forward neural network. The network was trained using spectrographs of different chords of a piano and it produced candidate chords. The network could output more than one hypothesis at each iteration, allowing the system to perform a parallel exploration of the solution space. Preliminary testing showed that the system had a tendency to misidentify octaves and make incorrect identification of note onsets, but these problems could potentially be solved by modifications to the signal processing system that feeds the blackboard system data and by refining the knowledge sources.

Rather than using a sequential scheduler to coordinate the blackboard system, Kashino used a Bayesian probability network (Kashino *et al*, 1995). Bayesian networks are well known for producing good results, despite noisy input or missing data. They are often used in implementing learning methods that trade off prior belief in a hypothesis against its agreement with current data. They therefore seem to be well suited to coordinating blackboard systems. There has not yet been any experimental research directly comparing the success of the sequential approach used by Martin to this Bayesian network approach, however. Aside from this important difference between Kashino's work and that of Martin, Kashino also used knowledge sources with information about stream segregation taken from research in human auditory scene analysis, as discussed above. Kashino also set up his system so that it could analyze signals containing more than one instrument. In order to accomplish this, he used knowledge sources programmed with the frequency components of different instruments played with different parameters.

In a later publication (Kashino and Hagita, 1996), Kashino suggested replacing the Bayesian network with a Markov Random Field hypothesis network. This allowed information to be integrated on a multiply connected hypothesis network, unlike the Bayesian network that only allowed singly connected networks. This made it possible to deal with two kinds of transition information within a single hypothesis network, namely chord transitions and note transitions. This approach was successful in correcting problems relating to misidentification of octaves and of instruments that plagued the previous system, although it did introduce some new errors. The new system performed achieved a recognition rate of 71.7% on a three-part arrangement of Auld Lang Syne, an overall improvement of roughly 10% over the old system.

Kashino later suggested a shift away from strict blackboard systems by performing more work in the front-end of the system and mathematically formalizing the work previously done by the knowledge sources (Kashino and Murase, 1998). Adaptive template matching was used in this new system. This system found the correlation between the output of a bank of filters arranged in parallel and a set of templates corresponding to particular notes played by particular instruments. This approach did achieve an average recognition rate of 88.5% on recordings of piano, violin and flute.

An alternative approach has been to take an input signal containing arbitrary instruments and extract information relating to only one of them. Some success has been achieved in extracting basslines in such a manner (Hainsworth and Macleod, 2001). High frequencies were filtered out of the signal and simple mathematical relations were used to trim hypotheses.

The work of Bello and Sandler (2000) and Marolt and Privosnik (2001) with feed-forward neural networks has shown some very interesting promise. It was decided to expand on their work here by examining in greater detail how systems of feed-forward neural networks could be applied to the transcription problem. We have forsaken the blackboard approach here not because it does not hold promise, but because we wanted to concentrate our energies on directly studying the applicability of neural networks. A future extension of this work could be to integrate part of the research here with a blackboard system.

Feed-forward neural networks consist of networks of sets of input units connected to hidden units, which are in turn connected to output units. Each input unit is linked by a weight to each hidden unit, and each hidden unit is linked by a weight to each output unit. Patterns of numbers are placed on each input unit, which are then multiplied by the weights and sent out to the hidden layer. The inputs to each hidden layer are summed, and this number is fed into an activation function, typically the sigmoidal function. These values are then propagated to the output units using the same procedure, with the result that each output unit ends up with a number between, typically between 0 and 1.

Series of training patterns can be fed to the input units of neural networks. A process of gradient decent in error space is performed to adjust the weights of the network so that the training data patterns produce (hopefully) a convergence towards the expected output at the output units. The hope is that the units will adapt to the training data so that they will be able to perform tasks such as approximate functions or perform pattern matching even when they are given data that they were not explicitly trained with. Neural networks are thus potentially useful at achieving difficult to describe mappings between input and output, which is why they are appropriate to apply to linking audio data to notes in a score.

3. Implementation

Our system was divided into two largely independent components: a signal processing front-end to extract frequency tracks and times of probable note onsets from audio files, and an array of neural networks to produce a score from this information.

Audio data for testing and training was produced by synthesizing MIDI recordings of clarinet and oboe performances with Apple's QuickTime player. Sets of training data consisted of chromatic scales (for the monophonic data) and all possible pairs of notes in a given range (for the duet data). Although this necessitated very long training sets, the result was that the network(s) had seen all possible inputs within the range of the synthesized sound, and—since the system processed each frame independently and was memoryless—should be well prepared to process test data. Note durations were kept short (an average of 300 ms) in order to try to force the system to recognize a note's transients as well as its steady state.

In order to synthesize audio data for testing, a variety of MIDI files were downloaded from the Internet. Several Schubert and Beethoven wind duets were employed to test the effectiveness of the transcription system described herein. Short segments (usually 10 to 15 seconds long) of varying styles and contrasting rhythms were selected and synthesized. The synthesized segments were then sent to the front-end for analysis. The irregular rhythms and melodic skips in the training data contrasted with the steady rhythms and intervals of the training data, thus providing "real-life" data that helped to provide insight as to the system's true effectiveness.

3.1 Implementation of Signal Processing front-end

The front-end of any transcription system is essential to its performance, as even the best performing back-end is impotent without reliable data from the front-end. As discussed above, two approaches, namely a sinusoidal and filterbank-based approach, have become generally accepted in the literature and have met some degree of success. The front-end discussed here therefore includes both of these.

The implementation of the front-end was performed entirely in MATLAB. Several toolkits from third-party sources were explored which extended MATLAB's functionality. Initial experiments were carried out with the SMT toolkit (Johnson, 2002). This set of tools allows audio data to be transformed in a number of ways and includes algorithms that can be used for note-onset detection. The STFT algorithm was initially used to generate spectral data, and peak-picking algorithms were used to extract the most prominent peaks. This data was then placed into an array, with new peaks added each frame. Unfortunately, this performed poorly in representing the evolution of each peak during transitions from frame to frame. A variety of window-sizes were employed to help smooth out the discontinuities from each frame (more reliable peaks, less variation in number), but the underlying problem still remained. Similar discouraging results occurred when the constant wavelet transform was used.

It was therefore decided to employ a more widely used and perhaps more robust method: SMS analysis (Serra, 1997). Spectral modeling synthesis is a set of techniques specifically geared towards the analysis of audio data. Meaningful data may be extracted from the signal in order to facilitate re-synthesis or further analysis. The algorithm models an input signal as a number of stable sinusoids (partials) plus a residual component. This implementation is advantageous for our purposes as the actual analysis detects partials by studying the time-varying spectral characteristics of the sounds. The resulting representation with time-varying sinusoids thus reflects the continually evolving nature of sound.

Prior to analysis, input audio was first converted to mono and downsampled to 11.025 kHz. Thus, frequencies as high as 5kHz could still be represented (allowing enough leeway for partials coming from a note produced in an instrument's upper range), while at the same time significantly reducing the amount of data. This was important, as a ten second audio file often took in excess of three minutes to analyze on a PowerBook G4 867 MHz machine.

After running several trial runs, it was discovered that a longer window-size greatly improved performance, even at the expense of time resolution. A window size of 2064 samples was found to be optimal, with a hop size of 512. Thus, each window was 47 milliseconds long, with a new analysis performed every 11.7 milliseconds.

A gammatone filterbank was also used to generate output from the front-end. The actual implementation of such a front-end was a grossly simplified version of similar correlation-based front-ends as described above, yet we were curious to see what sort of results might be gleaned from it. A gammatone filterbank was constructed in MATLAB using an algorithm from the *Auditory Toolbox* (Slaney, 1998). The centre frequency of each filter was defined to mimic similar filters of the basilar membrane. Sixty filters of a near-logarithmic spacing were deployed from 250 Hz up to the Nyquist frequency (5012.5 Hz, in this case). Because of the verbosity of the output, it was sampled every 256 samples to give a set of parameters every 5.8 milliseconds. Due to time constraints, we did not smooth the output of the filterbank by further processing with a hair-cell model or autocorrelation. This may have compromised our results.

One final note regarding all data produced by the front-end: a given partial's frequency and magnitude value were often two numbers with a large difference between them. This was a problem as wildly oscillating data such as this would adversely influence the performance of the neural networks in the back-end. To solve this problem, all input to the artificial neural network was scaled to values between 0 and 1.

3.2 Implementation of the Transcribing Back-end

The back-end of the network consisted of arrays of feed-forward neural networks with one hidden layer. The data from the front-end was fed into the networks in a variety of formats (see Table 1) and a variety of neural network architectures were implemented (see Table 2). In total, four different types of output from the front-end were fed into five different neural network architectures, for a total of twenty configurations. In order to provide further clarification, Table 3 shows the total number of networks and the number of input and output units needed for each configuration.

Experiments were conducted for each of these arrangements using a varying number of hidden nodes as well as different learning rates, momentums and ranges for initial randomly generated weights. Experiments were also conducted to find an appropriate number of partials to feed the networks. Although the system could take in an arbitrary of instruments, we only experimented with one or two instruments here.

A comparison between the first three network architectures in Table 2 made it possible to study the relative effectiveness of using separate networks for different notes and instruments as compared to using integrated networks. The last two network architectures in Table 2 determined pitch and the instrument separately. Although these latter two architectures could identify pitch and instrument when combined, they had the disadvantage of not being able to discriminate between simultaneous events. Nonetheless, these architectures were implemented in order to gain intermediate results if the first three architectures failed.

Input Format	Description
Sinusoidal tracks sorted by amplitude	Tracks with associated frequencies and amplitudes were generated from the audio data. These were then sorted based on amplitude and the frequency values were fed into the input units of the networks. The amplitude values were discarded after sorting.
Sinusoidal tracks sorted by frequency	Frequency tracks were generated from the audio data. These were then sorted and fed into the input units of the networks.
Track frequencies and amplitudes	Tracks with associated frequencies and amplitudes were generated from the audio data. These were then sorted based on amplitude and then both frequency and amplitude values were fed into the input units of the networks.
Gammatone filter-bank	The frequency components of the audio data were divided amongst frequency bins, the output of which were fed into the input units of the networks.

Table 1: Types of network input formats experimented with

Network Type	Description
One network per note per instrument[1]	A network was generated for every pitch of every instrument. Each network took in all of the available information for each frame and outputted a single value corresponding to its pitch and instrument. This allowed every possible note by every possible instrument to be considered independently by the system.
One network per instrument	One network was generated for each instrument. Each network took in all of the available information for each frame and had an output unit for every possible pitch in the specified range. This allowed the output pitch to be considered collectively by the networks while the instrument choice was considered independently.
Combined network	A single network took in all input from the front-end and had an output for each note on each instrument. This allowed pitch and instrument choice to be considered collectively.
Pitch only	A single network took in all input from the front-end and had an output unit for every pitch. No discrimination was made based on instrument.
Instrument playing only	A single network took in all input from the front-end and had an output unity for every instrument. Pitch was not identified.

Table 2: Network architectures experimented with

[1] The architecture was heavily influenced by the SONIC system (Martolt & Privosnik, 2001).

Network Type	Input Type	Number Nets	Input Units / Net	Output Units / Net
1 net / note / instrument	f sorted by A	I*R	B	1
	f sorted by f	I*R	B	1
	f and A	I*R	2*B	1
	filter banks	I*R	B	1
1 net / instrument	f sorted by A	I	B	R
	f sorted by f	I	B	R
	f and A	I	2*B	R
	filter banks	I	B	R
Combined network	f sorted by A	1	B	I*R
	f sorted by f	1	B	I*R
	f and A	1	2*B	I*R
	filter banks	1	B	I*R
Pitch only	f sorted by A	1	B	R
	f sorted by f	1	B	R
	f and A	1	2*B	R
	filter banks	1	B	R
Instrument playing only	f sorted by A	1	B	R
	f sorted by f	1	B	R
	f and A	1	2*B	R
	filter banks	1	B	R

Table 3: Network sizes. I = number of instruments, R = number of notes (i.e. range) and B = number of tracks (or number of bins in the case of the filterbank inputs). f = frequency and A = amplitude.

The networks produced outputs for every frame. These were collected into arrays indexed by frame and pitch (which was derived from which output unit a value came from) and thresholded so that all outputs below 0.5 were counted as off and all values above 0.5 were counted as on. All consecutive ons were collected into groups, with each group counting as a note with its duration determined by the number of consecutive ons multiplied by the frame length.

The system had produced a score at this point. However, this score was somewhat noisy, so the data was then cleaned before a final score was produced. The biggest problem was that occasionally there might be one or two frames that produced offs during a sustained note, or a few frames that produced ones during rests. This resulted in rapid staccato notes during a sustained note or a few staccato notes during a rest. In order to counteract this, the score was smoothed so that isolated ons in sequences of offs or isolated offs in sequences of ons were inverted to produce steady notes or rests. Although, this would cause the system to miss very rapid notes that only lasted a frame, the frame size was small enough that this scenario was unlikely. Also, since the instruments dealt with here (clarinet and oboe) could only produce one note at a time, only the on with the highest output level was counted if the networks indicated that the same instrument should be producing more than one note at a time.

Finally, a GUI was constructed that enabled the user to create tailored arrays of networks, train them and pass test data through trained networks to generate scores. These final transcriptions could be saved as MIDI files or could be viewed as raw output on a frame by frame basis.

4. Results

There were some early problems with training the networks, the foremost being that the networks had a tendency to converge to always being off (i.e. no notes were detected). Our first suspicion was that this was due to the tendencies of tracks to switch positions (remember that they were sorted based on frequency or amplitude) due to small oscillations when they had close amplitudes or frequencies. This had the potential of being very problematic because neural networks need to see at least somewhat consistent patterns to converge during training. A network can get confused if a given set of inputs corresponds to one expected output for a given input set but another expected output when the same set of inputs is seen at a later time.

In order to counteract the problem of switching tracks, we decided to increase the frame size to smooth out this problem. Although this was successful in preventing the switching tracks, the networks still had problems converging. We experimented with further techniques such as normalizing and scaling the input to values between 0 and 1 to make the different input units start off on a more or less equal footing and make the networks' initial search of the error space easier. Although these techniques did help, the networks still often converged to always being off.

Our next suspicion was that, since each note was off during most of the training data, particularly for larger pitch ranges, the networks were seeing many offs in a row during training and forgetting the brief ons, thus learning to always output offs. The networks were trained with chromatic scales where each note would be on for only a brief period, but then off during all other notes. Since there were far more offs than ons, especially for large numbers of pitches, it was theorized that these extended periods of uninterrupted offs during training caused the networks to effectively forget the little bit of training that they received during the brief on periods. In order to solve this problem, we randomized the positions of each frame in the training data so that the ons would be spread throughout each epoch rather than being clustered together. This technique was successful in causing the networks to begin converging, so it appears that our hypothesis was likely correct.

Although it was found that very good performance could be achieved eventually, this required significant training, on the order of at least 100 000 epochs. Since it took roughly one day per 50 000 epochs to train a one octave network for two instruments on a modern PC, this limited the number of experiments that we could carry out on different types of networks. It also raised questions about the scalability of the system if training were to be limited to basic PCs. Since a greater range and greater number of instruments would eventually be desired, scaling this system would require access to a supercomputer.

Results are still preliminary at this stage, as there was not enough time to fully test many of our network architectures and input formats. Due to these time constraints, we chose to focus on the most promising approach, namely the 1 net / note / instrument architecture with the sinusoidal tracks sorted by amplitude input. All twenty possible configurations were compared after 5000 training epochs, and this one performed the best. This does not mean that the other approaches should not be considered in future research, however, since proper training took at least 100 000 epochs, and an approach that performed poorly at first could end up performing better in the long run. More experimentation is therefore needed to find the best approach.

All implementations based on a filterbank front-end data were slow to train and showed no viable results. This is most likely due to the oversimplified data produced from the front-end module; further processing of the data to more closely mimic "human perception" would likely improve results.

Sinusoidal tracks sorted by frequency also a performed poorly. Training was abandoned early on as the networks did not converge on this type of input. Networks based on both frequency and amplitude were also abandoned due to the length of time required to efficiently train them. This latter implementation has a great deal of potential because of the verbosity of the input data, but this same verbosity has a detrimental effect on training time.

The architecture that utilized one network for every note of every instrument and operated on sinusoidal track data sorted by amplitude performed the best in early trials. It was thus was the one that was selected for extended training, for a total of 100 000 epochs. This system succeeded in correctly transcribing 97% of the 2-voice frames and 99% of the monophonic frames.

This performance of the system was excellent in regards to pitch. Almost every note was detected, almost no false notes were detected and the pitch identified was almost always correct. Unfortunately, even the relatively small number of erroneously transcribed sample-frames resulted in considerable rhythmic irregularities in the transcriptions, with notes often shorter than they should be, or occasionally slightly longer. This was likely due to two factors: the networks were likely not always able to associate initial transients with pitches and the fairly long frame size may have lead to an excessively quantized and uneven rhythm. The former problem could possibly be solved by more training and the latter could be solved by decreasing the frame size.

This was not done here because both of these solutions would have increased training time, but they could certainly improve results given more time and processing power. Future use of onset-detection information could also help to clean these problems up.

The error rates discussed above give the percentage of frames that were correctly associated with a pitch or a silence. Although these do give an indication of the performance of the system, a better metric would consider the actual notes produced in the score itself, including rhythmic irregularities. One such metric would be to measure the system's efficiency with the Dixon formula (Dixon, 2000). Dixon proposes an evaluation method based on the number of paired notes, *N*, false positives, *FP*, and the number of false negatives, *FN*. An incorrectly identified note is both false positive and a false negative in this model. A score, *S*, may be obtained with the following formula:

$$S = \frac{N}{N + FP + FN}$$

Although we did not have time to implement this metric, it would be useful in better judging the performance of the system in the future.

5. Conclusions

The system performed very well in correctly identifying pitches and avoiding missed notes and false notes. The rhythm was choppy, but this could likely be improved with further time and processing power and by using the note onsets provided by the front end. Our system performed particularly well in regards to detecting short notes and avoiding the misidentification of octaves, two problems that have plagued many other systems. The system also avoided using high-level music theory to improve results as some other systems do, so no stylistic assumptions were made here.

This study was useful in examining different kinds of network architectures that could be used for transcription, although more experimentation is needed to take advantage of the groundwork laid here. We found that using scales to train the networks was effective, as long as the frames corresponding to each note were spread throughout the training data by randomizing their position.

The question to examine in future research is how well these techniques can scale to the general problem of transcription. The excellent performance of this system bodes well, although it is clear that access to significant computing power will be necessary to expand it. Experiments could be conducted to see how audio from different synthesizers performs on this system. If this is successful, the next step would be to test the system with larger orchestrations, such as trios or quartets, and with instruments that can play more than one note at a time, such as pianos. Finally, the system should be tested on unsynthesized audio.

Training could be difficult with unsynthesized audio, since great attention must be paid to producing a model score that matches the audio exactly. One approach that might simplify this problem would be to train the data using a MIDI file that is translated into audio using a high quality sampling synthesizer, since the MIDI file could then be used to automatically generate the model score. Audio test sets could then be used on the trained networks.

6. Bibliography

Bello, J. P., G. Monti, and M. Sandler. 2000. Techniques for automatic music transcription. *Proceedings of the International Symposium on Music Information Retrieval.*

Bello, J. P., and M. B. Sandler. 2000. Blackboard system and top-down processing for the transcription of simple polyphonic music. *Proceedings of the COST G-6 Conference on Digital Audio Effects.*

Bello, J. P., L. Daudet, and M. B. Sandler. 2002. Time-domain polyphonic transcription using self-generating databases. *Proceedings of the 112th Convention of the Audio Engineering Society*.

Dixon, Simon. 2000. Extraction of Musical Performance Parameters from Audio Data. *Proceedings of the First IEEE Pacific-Rim Conference on Multimedia.*

Engelmore, R. S., and A. J. Morgan. 1988. *Blackboard Systems.* Addison-Wesley Publishing.

Goto, M. 2001. A predominant-F0 estimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models. *IEEE International Conference on Acoustics, Speech and Signal Processing.* 3365-8.

Hainsworth, S. W. and M. D. Macleod. 2001. Automatic bass line transcription from polyphonic music. *Proceedings of the International Computer Music Conference.*

Johnson, K. 2002. *The spectral modeling toolbox: A sound analysis/synthesis system*. Master's Thesis. Dartmouth College.

Kashino, K., K. Nakadia, T Kinoshita and H. Tanaka. 1995. Application of Bayesian probability network to music scene analysis. *Proceedings of the International Joint Conference on AI, CASA workshop.* 52-9.

Kashino, K. and Norihiro Hagita. 1996. A music scene analysis system with the MRF-based information integration scheme. *Proceedings of the International Conference on Pattern Recognition.* 725-9.

Kashino, K. and Hiroshi Murase. 1998. Music recognition using note transition context. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.* 3593-3596.

Klapuri, A. 2001. Multipitch estimation and sound separation by the spectral smoothness principle. *IEEE International Conference on Acoustics, Speecha and Signal Processing.* 3381-4.

Kröse, B., and P. van der Smagt. 1994. *An Introduction to Neural Networks*. Available at www.robotic.dlr.de/Smagt/books/neuro-intro.ps.gz.

Marolt, M, and M. Privosnik. 2001. SONIC: A system for transcription of piano music. *Proceedings of the Workshop on Current Research Directions in Computer Music.* WSES Press.

Martin, K. D. 1996a. A blackboard system for automatic transcription of simple polyphonic music. *M.I.T. Media Lab Perceptual Computing Technical Report #385.*

Martin, K. D. 1996b. Automatic transcription of simple polyphonic music: Robust front-end processing. *M.I.T. Media Lab Perceptual Computing Technical Report #399.*

Miwa, T., Y. Takokoro, and T. Saito. 2000. Musical pitch estimation and discrimination of musical instruments using comb filters for transcription. *42nd Midwest Symposium on Circuits and Systems.* 105-8.

Pickens, J., et al. 2002. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. *Proceedings of the International Symposium on Music Information Retrieval.* 140-9.

Russell, S. and P. Norvig. 1995. *ArtificialIntelligence: A Modern Approach.* Prentice-Hall Inc.

Sano, H., and B. K. Jenkins. 1989. A neural network model for pitch perception. *Computer Music Journal* 13 (3).

Serra, X. 1997. *Musical Sound Modeling with Sinusoids Plus Noise. In Musical Signal Processing.* C. Roads, S. Pope, A. Picialli, and G. De Poli eds. Swets & Zeitlinger Publishers.

Slaney, M. 1998. *Auditory Toolbox version 2.* Apple Technical Report.