Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition Letters 26 (2005) 1809–1818
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Polyphonic monotimbral music transcription using dynamic networks

Antonio Pertusa, José M. Iñesta *

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Apartado de correos, 99 Alicante 03080, Spain

Available online 14 April 2005

Abstract

The automatic extraction of the notes that were played in a digital musical signal (automatic music transcription) is an open problem. A number of techniques have been applied to solve it without concluding results. The monotimbral polyphonic version of the problem is posed here: a single instrument has been played and more than one note can sound at the same time. This work tries to approach it through the identification of the pattern of a given instrument in the frequency domain. This is achieved using time-delay neural networks that are fed with the band-grouped spectrogram of a polyphonic monotimbral music recording. The use of a learning scheme based on examples like neural networks permits our system to avoid the use of an auditory model to approach this problem. A number of issues have to be faced to have a robust and powerful system, but promising results using synthesized instruments are presented.

1. Introduction

Tone perception is a complex phenomenon. The human ear can detect musical tones even in presence of noise. We can also hear a number of simultaneous tones and detect subtle but expressive tonal deviations (vibrato, microtonal intervals, ...). A problem related to this ability in computer science is the automatic score extraction from digitized music or, for short, music transcription.

Music transcription can be defined as the act of listening to a piece of music and writing down music notation for the notes that make up the piece (Martin, 1996). The automatic transcription of monophonic signals (only one note playing simultaneously) is a largely studied problem. Several algorithms have been proposed that are reliable, commercially applicable, and operate in real time. Nevertheless, automatic polyphonic music transcription is an open research problem, because not even the perceptual mechanisms involved in the isolation of different notes and instruments and their insertion in the corresponding musical phrases are clear.

* Corresponding author. Fax: +34 96 5909326.
E-mail addresses: pertusa@dlsi.ua.es (A. Pertusa), inesta@dlsi.ua.es (J.M. Iñesta).

0167-8655/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2005.03.001

The first polyphonic transcription system dates back to the 1970s, when Moorer (1975) built a system for transcribing two-voice compositions. Recent state-of-the-art in music transcription has been discussed by Klapuri (1998). There is still no method to solve completely this problem, but there are mainly two pitch detection method groups: time domain methods and frequency domain methods.

Time domain methods are useful for monophonic transcription, but they have shown poorer results for polyphonic transcription than frequency methods. Most of time domain methods are based on autocorrelation (Brown and Zhang, 1991), time domain peak and valley measurements, or zero-crossing measurements. Frequency domain methods look for fundamental frequencies, whose harmonics best explain the partials[1] in a signal (Hess, 1983).

In this paper, we present a frequency domain method based on dynamic neural networks. Connectionist approaches have been used in music for a long time (Todd and Loy, 1991) and they have been also used for music transcription (Shuttleworth and Wilson, 1993), recently with polyphonic piano music (Marolt, 2001), and they present a good alternative in building transcription systems. In the latter work, Marolt's system (SONIC) tracks groups of partials in piano music with 76 adaptive time-delay neural networks (TDNN) and postprocessing, obtaining good results. TDNN have also been used successfully earlier in speech recognition problems (Hertz et al., 1991; Lang et al., 1990).

Good results have been obtained for polyphonic transcription using human auditory models and signal processing methods (Tolonen and Karjalainen, 2000; Klapuri, 2003). In the present work, on the contrary, we aim to avoid any kind of auditory model or signal processing method. Our objective is to discover, in a simplified version of the problem, whether a learning algorithm, with the only input of spectral bands, can learn to detect the notes that are sounding in a polyphonic melody. A dynamic neural network is utilized for detecting notes sounding at each time. No audio or spectrum processing motivated by an auditory model is performed before being entered into the recognition system.

TDNN are considered as non-recurrent dynamic networks (Hush and Horne, 1993), although essentially are like static nets traversing temporal series. This kind of nets are able to model systems where the output $y[t]$ has a dependence of a limited time interval in the input $u[t]$:

$$y(t) = F[u(t-m), \ldots, u(t), \ldots, u(t+n)]. \quad (1)$$

With this network, time series can be processed as a collection of static input–output patterns, related in short-term as a function of the width of the input window. Due to the absence of feedback, the net architecture is the same as that of a multilayer perceptron and it can be trained by a standard backpropagation algorithm (Rumelhart et al., 1986).

The spectral pattern of a given sound signal $s(t)$ is the energy distribution that can be found in the constituent partial frequencies of its spectrum. This pattern is the most important parameter for characterizing the musical timbre.[2]

In this work, the music transcription problem is posed through the identification of the pattern of a target instrument, using a connectionist dynamic algorithm, like time-delay neural networks, previously trained with spectrograms computed from polyphonic melodies played by it. Only tuned sound sources, those that produce a musical pitch, are considered, putting aside those that produce noises or highly inharmonic sounds. Also, smooth envelope timbres (i.e. waveshapes with a nearly stable amplitude from the beginning to the end of each note) without volume changes are considered.

To achieve this goal, input and output pairs are needed. They are formed by the band-grouped spectra of the sound and the information about which notes are producing them. Each spectrum

[1] A "partial" is any of the frequencies in a spectrum, being "harmonic" those multiples of a privileged frequency called fundamental that provides the pitch of the sounding note.

[2] In acoustics, timbre is defined as the quality of a sound that permits to perceive it as different from other sounds of equal pitch and amplitude.

is grouped into 1/12 octave bands centered in the musical pitches. This simplified representation keeps the main structure of the spectral pattern (if source is tuned), and makes the problem easier for handling with a neural network. For each time available in the training set, t_i, the input is a set of bands, $\{S(f, t_{i+j})\}$ for $j \in [-m, +n]$, being m and n the number of spectrum windows considered before and after t_i. The output consists of a binary vector $v(t_i)$ coding the set of notes that are active at that moment in order to produce those spectra.

The working hypothesis is that, after the learning phase of the structure of a sound source, the net will be able to detect the notes as occurrences of that pattern in the spectrogram of a melody produced by that source.

2. Methodology

2.1. Construction of the input–output pairs

The training set is formed by pairs $\{\{S(f, t_{i+j}), j \in [-m, +n]\}, v(t_i)\}$, being $S(f, t)$ the spectrum bands obtained from the target melody at a given time t and $v(t)$ a binary vector representing the notes that are sounding. The spectrum frequencies are grouped into b bands in a logarithmic scale of a twelfth of octave (a halftone), centered in the well-tempered scale frequencies. For the output, we need a set of music scores in digital format and to synthesize sounds according to the instructions in them, in such a way that the correspondence between the spectrum and the set of notes that have produced it can be known at every moment. For this, MIDI files were used as digital scores and a software synthesizer developed by the Medialab at MIT named Csound (Vercoe, 1991; Boulanger, 1999) was used to generate the music files.

First we get into the details of the input data construction ($S(f, t)$) and then the training outputs ($v(t)$) are presented.

2.1.1. Input data

From each MIDI sequence, a digital audio file was synthesized and its short-time Fourier transform (STFT) was computed, providing its spectrogram. For it, a Hanning window was utilized, described at instant τ by this expression:

$$w(\tau) = \frac{1}{2}\left(1 - \cos\frac{2\pi\tau}{N}\right), \tag{2}$$

where $N = 2048$ is the number of samples in the window. Also an overlapping percentage of $O = 50\%$ was applied in order to keep the spectral information at both ends of the window.

The time resolution for the spectral analysis, $\Delta t = t_{i+1} - t_i$, can be calculated as

$$\Delta t = \frac{(100 - O)N}{100 f_s}. \tag{3}$$

In order to have less frequency bands and less computational load, divisors of the sampling rate can be used, although this limits the useful frequency range. The original sampling rate of the digital sound files was $f_s = 44{,}100$ Hz, nevertheless an operational sampling rate of 44,100/2 = 22,050 Hz was used. Thus, the highest possible frequency is $f_s/2 = 11{,}025$ Hz, which is high enough to cover the range of useful pitches. With this value, Eq. (3) yields $\Delta t = 46.4$ ms. This time establishes the precision in time for detecting the onset and the end of the notes.

With the parameter values described, the STFT provides 1024 frequencies with a spectral resolution of 10.77 Hz that are grouped into b bands in a logarithmic scale ranging from 50 Hz (for a pitch close to $G\sharp_0$—G sharp of octave 0—) to 10,600 Hz (pitch F_8), almost eight octaves. This way, $b = 94$ spectral bands are obtained that correspond to the 94 notes in that frequency range, and they will be provided to each of the 94 neurons in the net input layer.

The amplitudes of the bands in the spectra are obtained in dB as attenuations from the maximum amplitude. The dynamic range of the considered digital audio signal is 96 dB, provided the 16 bits resolution utilized. In order to remove noise and emphasize the important frequency bands in each window position, a low level threshold, θ, is applied in such a way that for each band f_k, if $S(f_k, t_i) < \theta$ then $S(f_k, t_i) = \theta$. See Fig. 1 for a picture of this scheme. This threshold was empirically established at $\theta = -45$ dB.

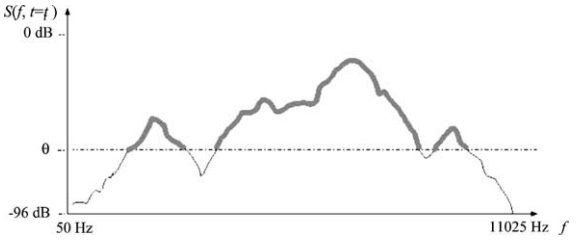

Fig. 1. A low level threshold is applied to remove noise.

Usually, a note onset is not centered in the STFT window, so the bell-shape of the window affects the amplitude and some amount of energy can be lost if the note starts or ends off-centered in the window. A dynamic net as TDNN becomes useful to minimize this problem. The overlapping adjacent positions of the spectrogram provide this context information to the net. For each window position considered, b new input units are added to the net, so the total number of input neurons is $b \times (n + m + 1)$. See Fig. 2 for a scheme of this architecture.

2.1.2. Output data

The net output layer is composed of 94 neurons (see Fig. 2), one for each possible note (and the same number of spectral bands at each input). Therefore, a symbolic output is provided with as many neurons as notes are in the valid range. The output is coded in such a way that an activation value of $y(t_i, k) = 1$ for a particular unit k means that the kth note is active at that time, and 0 means that the note is not active. The training vectors are, therefore, $v(t_i) \in [0, 1]^{94}$. Usually the number of zeros is much larger than that of ones, because only a small subset of possible notes are active at each moment.

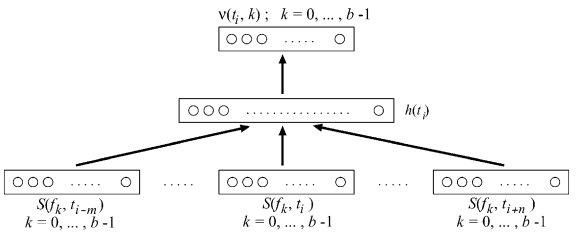

Fig. 2. Network architecture and data supplied during training. The arrows represent full connection between layers.

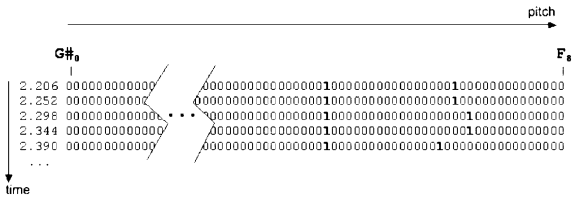

Fig. 3. Binary digital piano-roll coding in each row the notes that are active (1's) at each moment when the spectrogram is computed.

The series of vectors $v(t_i)$, $t_i = 1,2,\ldots$ has been named binary digital piano-roll (BDP). A brief example can be observed in Fig. 3. The vectors for each time are the training outputs shown to the net during the training phase, while the corresponding band values are presented to the input.

Each BDP is computed from a given MIDI file, according to the notes that are active at the times where the windows of the STFT are centered.

2.2. Network parameters

A time-delay neural network has been used, trained with the standard backpropagation algorithm. The network is implemented with bias and without momentum. A standard sigmoid has been used as transfer function. Before providing the attenuations to the net input they are normalized to the interval $[-1, +1]$, being the -1 value assigned to the maximum attenuation (θ dB). The value +1 is assigned to the attenuation of 0 dB. This way, the input data $S(f, t) \in [\theta, 0]$, are mapped into the range $[-1, +1]$ for the network input through this function:

$$f(x) = \frac{1}{\theta/2}(\theta + x) - 1. \tag{4}$$

A number of parameter values are free in any net. Some of them have a special interest in this work.

Number of spectrum windows at the input $(n + m + 1)$. The upper limit is conditioned by the fact that it has no sense to have a number of windows so large that spectra from different note subsets appear frequently together at the net input, causing confusion both in training and recognition. Moreover, the computational cost depends

on this number. A good contextual information is desirable but not too much.

Activation threshold (α). The output value for the neurons are $y_k(t) \in [-1, +1]$. The final value to decide whether a note has been detected at time t_i is computed as 1 if $y_k(t_i) > \alpha$ and 0 otherwise. The lower is α, the more likely a note is activated. This value controls the sensitivity of the net.

These are the net most influent parameters, while others concerning the training, like weight initialization, number of hidden neurons, initial learning rate, etc. have shown to be less important. Different experiments were carried out varying them and the results did not vary importantly. After some initial tests, a number of hidden neurons of 100 has proven to be a good choice for that parameter, so all the results presented have been obtained with this number of hidden neurons.

2.3. Success assessment

A measure of the quality of the performance is needed. We will assess that quality at two different levels: (1) considering the detections at each window position t_i of the spectrogram. The output at this level will be named "event detection"; and (2) considering notes as series of consecutive event detections along time. The output at this level will be named "note detection".

At each time t_i, the output neuron activations, $y(t_i)$, are compared to the vector $v(t_i)$. A successful detection occurs when $y(t_i,k) = v(t_i,k) = 1$ for a given output neuron k. A false positive is produced when $y(t_i,k) = 1$ and $v(t_i,k) = 0$ (something has been detected but it was not actually a note), and a false negative is produced when $y(t_i,k) = 0$ and $v(t_i,k) = 1$ (something has been missed). These events are counted for each experiment, and the sums of correct detections, Σ_{OK}, false positives Σ_+, and false negatives Σ_- are computed. Using all these quantities, the success rate for detection in percentage is defined as:

$$\sigma = \frac{100\Sigma_{OK}}{\Sigma_{OK} + \Sigma_- + \Sigma_+}. \quad (5)$$

With respect to notes, we have studied the output produced according to the criteria described above and the sequences of event detections have been analysed in such a way that a false positive note is detected when a series of consecutive false positive events is found surrounded by silences. A false negative note is defined as a sequence of false negative events surrounded by silences, and any other sequence of consecutive event detections (without silences inside) is considered as a successfully detected note. Eq. (5) is also utilized to quantify the note detection success.

3. Results

3.1. About the data

Polyphonic tracks of MIDI files were selected, containing chords and scales, trying to have different number of note combinations sounding at different times in order to have enough variety of situations in the training set. 2377 different chords appeared in it. The number of spectrum samples was 31,680 (around 25 min of music).

The training algorithm converges quickly (tens of epochs) and each epoch takes about 15 s in a 1.3 GHz PC.

In order to test the ability of the net with waves of different spectral complexity, the experiments have been carried out using different waveshapes for training different nets. The limitations for acoustical acquisition from real instruments played live by musicians and the need of an exact timing of the emitted notes have conditioned our decision for constructing these sounds using virtual synthesis models.

A number of different timbres were considered. Some of them were synthetic waveshapes (sounds that cannot be found in any real sound source). In addition to those artificial waveshapes, real instrument timbres were utilized. They were synthesized through physical modelling algorithms using Csound. These methods of sound synthesis use a model of the instrument sound production instead of a model of the sound itself.

After performing some initial experiments with this set of sounds, two timbres from the first class and two from the second one were selected as representatives of the behaviour of our system with them. Those timbres were:

Sinusoidal waveshape. This is the simplest wave that can be analyzed. All its spectrum energy is concentrated in a single partial. This way, the frequency band of maximum amplitude corresponds to the frequency of the emitted note.

Sawtooth waveshape. This wave contains all the harmonic partials with amplitudes proportional to $1/p$, being p the number of partial. It presents high complexity because the amplitude of its partials decrease slowly. We have used only the first 10 partials to synthesize this timbre.

Clarinet waveshape. We wanted to use an imitation of an acoustic instrument. Different ones that had the desired properties of stability in volume were tested and finally a physical model of a clarinet that produces good imitating synthesis was selected.

Hammond organ waveshape. We also tried to include a timbre from an electronic instrument. A Hammond organ was selected. This is an instrument that produces sound through a mechanism based on electromagnetic induction. Here, this mechanism has been simulated by software with the Csound synthesizer.

3.2. Network parameter tuning

According to the size of the input context, the best results were obtained with one previous spectrogram window and zero posterior windows. Anyway, these results have not been much better than those obtained with one window at each side, or even 2 + 1 or 1 + 2. The detection was consistently worse with 2 + 2 contexts and larger ones. It was also interesting to observe that the success rate was clearly worse when no context was considered (around 20% less than with any of the other non-zero contexts tested).

In order to test the influence of the activation threshold, α, some values have been tested. High values (namely, above zero) have shown to be too high and a lot of small good activations were missed. As the value of α gets low, the sensitivity and precision of the net increases. Values of $\alpha \in [-0.8, -0.7]$ have shown to perform the best.

Once these parameters have been tuned and their consistency for the different waveshapes tested (within a small range of variation), a number of cross-validation experiments were performed in order to assess the capability of the net to carry out this task with different timbres.

For training, the whole data set was divided into four parts and four sub-experiments have been made with 3/4 of the set for training and 1/4 of the set for test. The presented results are those obtained by averaging the four sub-experiments carried out on each data subset.

3.3. Recognition results

In Table 1, the best results of note and event detection for the timbres described in Section 3.1 are presented. As expected, due to the simplicity of its spectral pattern, the sinusoidal waveshape has provided the best results (around 94% for events and 95% for notes). Most of the event detection errors were false negatives in the onsets and ends of the notes and the majority of the errors in note detection corresponded to notes of less than 0.1 s of duration (just one or two window positions in the spectrogram) that were not detected.

For the sawtooth timbre, the success results are lower due to the higher number of harmonics of its spectral pattern. Around 92% for events and also for notes were obtained. Again, most of the false negatives occurred in very short notes.

Concerning to the instrument timbres, both for the clarinet and the Hammond organ the results were comparable to those obtained for the synthetic waveshapes, giving values around 92% for notes and ranging from 90% to 92% for events.

These results with the clarinet and the Hammond suggest that the methodology can be applied to other instruments characterized by being nearly stable in time along the duration of each note. This applies, for example, to the wind and bowed

Table 1
Detection results (σ in Eq. (5)) in percentages

	Sine (%)	Sawtooth (%)	Clarinet (%)	Hammond (%)
Events	93.6	92.1	92.2	90.7
Notes	95.0	91.7	91.7	91.9

strings instrument families. For more evolving timbres, like for example plucked strings, more context information is needed.

The errors have been analyzed considering note length, pitch and number of training samples. Errors produced by short notes (less than 0.1 s) represent the 31% of total errors. Using a time resolution $\Delta t = 46.4$ ms, these notes extend along one or two events. Since most of the false negatives occurred in the onsets and ends of the notes, it is easy to understand that these very short notes can be missed by the net.

Most of pitch errors correspond to very high (higher than C_7) and very low (lower than C_3) pitches. In Fig. 4(top) the recognition percentage is represented as a function of the pitch. Note that

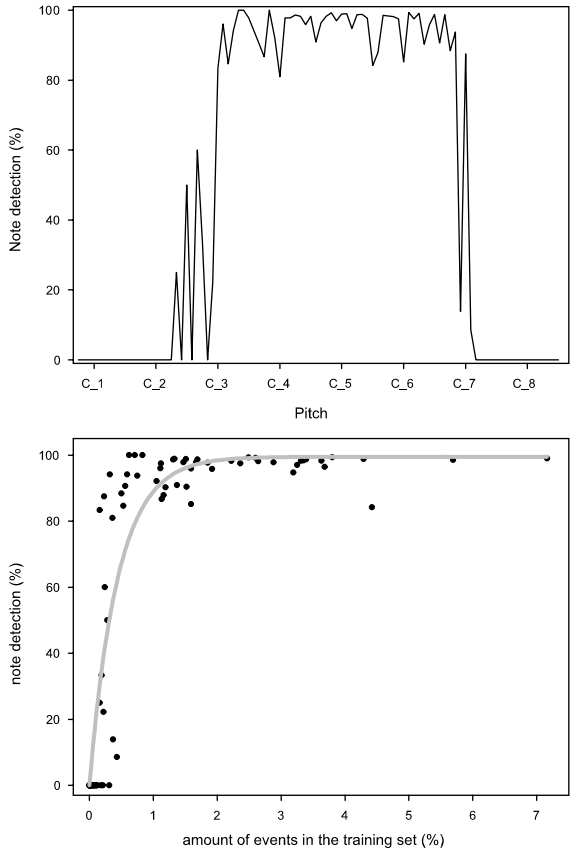

Fig. 4. Top: recognition rate as a function of pitch. Bottom: correlation between recognition rates for each pitch and the amount of events in the training set for that pitch.

the system performs well in the central band of pitches. This is motivated by two main problems that are discussed next.

Firstly, very low frequencies are harder to analyse due to the linear nature in frequency of the Fourier analysis, in contrast to the logarithmic nature of pitch. When constructing the higher bands, tens or even hundreds of frequency bins are provided by the STFT, but the lowest pitches use just one or few bins. This fact makes low pitches to appear fuzzy in the spectrogram. On the other hand, the highest harmonics are very close to the high limit in frequency in the digital domain (the Nyquist frequency = $f_s/2$) and this motivates a reflection of some of their partials (aliasing effect) that introduces confusion in the training.

Secondly, it has to be noted that the training set was composed of actual musical data, and therefore the usual octaves in which the music concentrates are the central band of pitches represented in Fig. 4(top). In fact, there exists a clear correlation of recognition success for a given pitch to the amount of events in the training set for that pitch. In Fig. 4(bottom) each dot represents a single pitch. Abcises represent the amount of data for that pitch in the training set and the ordinates represent the recognition percentage. An exponential curve has been adjusted to the data showing the clear non-linear correlation between training data and performance.

3.4. Changing waveshapes for detection

The results for the different waveshapes considered were very similar. It seems that the performance is not highly conditioned by the selected timbre. Thus, the doubt arose about how specific the net weights were for the different timbres considered. For this, to test the net performance, melodies played with a given waveform were presented to a net trained with band-grouped spectrograms from a different waveform. Nets for the four waves were trained and tested. The event and note detection results are displayed in Tables 2 and 3, respectively.

The success rates range from 8% to 70% for transcription of sounds different from those used

Table 2
Event cross-detection results

	Sine (%)	Sawtooth (%)	Clarinet (%)	Hammond (%)
Sine	93.6	56.6	70.2	29.6
Sawtooth	48.0	92.1	68.8	26.3
Clarinet	46.0	62.6	92.2	34.2
Hammond	8.3	16.9	14.4	90.7

Rows correspond to training timbres and columns to testing timbres.

Table 3
Note cross-detection results

	Sine (%)	Sawtooth (%)	Clarinet (%)	Hammond (%)
Sine	95.0	45.6	60.5	27.0
Sawtooth	50.7	91.7	65.4	28.6
Clarinet	56.8	55.5	91.7	33.7
Hammond	8.9	16.4	14.0	91.9

Rows correspond to training timbres and columns to testing timbres.

to train the net, so they are clearly worse for this experiment, showing the specificity of the nets.

3.5. Evolution in time for note detection

A graphical study of the net output activations along time has been carried out in order to analyze the kind of errors produced when compared to the desired output (see Fig. 5). In the plot, the active neurons at each moment may have either a 'o' if they successfully detected an event, or a '+' if the activation corresponded to a false positive event. If there was no activation for a given neuron at a time where the corresponding note was actually sounding, a '–' was displayed, corresponding to a false negative. Where neither notes nor activations appeared, nothing was displayed.

The example melody of Fig. 5 was neither in the training set nor in the recognition set. It was downloaded from the Internet and synthesized using the clarinet timbre. It is a difficult melody to be transcribed because tempo is 120 beats per minute and, therefore, quarter notes last for 0.5 s, eighth notes last for 0.25 s and 0.125 s for sixteenths. The total duration of the melody is 7.5 s. The event detection rate was 94.3%, the proportion of false

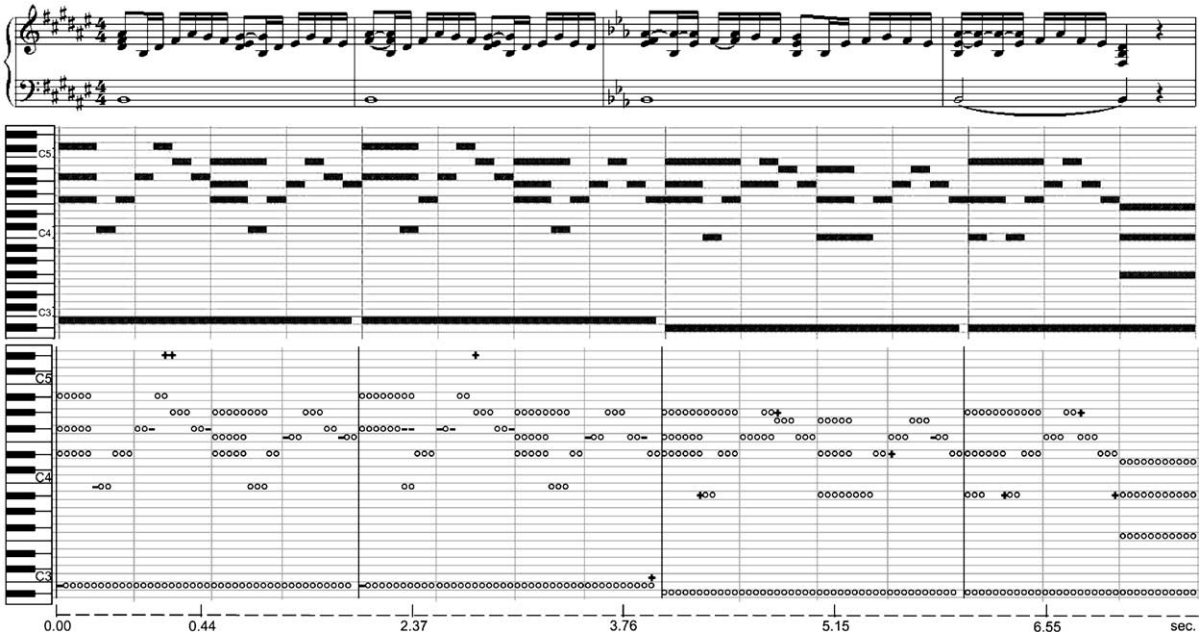

Fig. 5. Evolution in time of the note detection for a given melody using the clarinet timbre. Top: the original score; center: the melody as displayed in a sequencer piano-roll; down: the piano-roll obtained from the net output compared with the original piano-roll. Notation: 'o': successfully detected events, '+': false positives, and '–': false negatives.

positives to the number of events detected was 2.3%, and for false negatives was 3.4%. For notes, all of them were detected and just three very short false positive notes appeared.

Note that most of the errors were produced in the transition events between silence and sound or vice versa for some notes. This is due to the time resolution that causes the presence of energy when a note is not coded in the BDP as sounding or a lack of energy when it is already sounding according to the BDP. Also, interferences among frequency partials lead to errors like it seems to happen in the first beat of the second measure, where a new short note causes a false negative of a higher note already sounding.

4. Discussion and conclusions

This work has tested the feasibility of an approach based on time-delay neural networks for polyphonic monotimbral music transcription. A TDNN is fed with 1/12 octave band spectrograms of melodies played from MIDI files by both synthetic waveshapes and synthesized real instruments using a physical modelling virtual synthesizer.

The results suggest that the neural network is learning the pattern of the timbre and then is able to find complex mixtures of it in the spectrograms. The detection success was around a 92% in average and was somewhat independent of the complexity of the pattern. This seems to be one of the points in favour of the nets. Other pattern recognition approaches (like k-nearest neighbours or a 94 Bayesian ensembles) were tested and their performance worsen as the complexity of the spectral pattern increased. For example, a 90% was achieved for sine waveshape by simply thresholding the bands spectrogram looking for the fundamental frequency of each note. This same procedure provided only a 23% for the Hammond organ.

When the test waveshape was different from that used to train the net, the recognition rate decreased dramatically, showing the high specialization of the net.

Errors concentrated in very low-pitched notes, where the spectrogram provides less precision, and in very high notes, where their higher partials are folded by the Nyquist frequency, distorting their spectra. Also, this success for the central band of pitches is due to the higher presence of notes of these pitches in the training set. This suggests that increasing the size and variety of the training set would improve the results.

Most of the errors are concentrated in the transitions, at both ends of the note activations. This kind of situations can be solved applying a post-processing stage over the net outputs along time. In a music score not every note onset is equally probable at each moment. The onsets and ends of the notes occur in times that are conditioned by the musical tempo, that determines the position in time for the rhythm beats, so a note in a score is more likely to start in a multiple of the beat duration (quarter note) or some fractions of it (eighth note, sixteenth note, etc.). The procedure that establishes tight temporal constraints to the duration, starting and ending points of the notes is usually named quantization. From the tempo value (that can be extracted from the signal) a set of preferred points in time can be set to assign beginnings and endings of notes. This transformation from STFT timing to musical timing should correct some of these errors.

False positive and negative notes are harder to prevent and it should be done using a musical model. Using stochastic models, a probability can be assigned to each note in order to remove those that are little probable. For example, in a melodic line it is very unlikely that a non-diatonic note two octaves higher or lower than its neighbours appears.

The capability of a net trained with a given timbre for transcribing audio generated by a different, but similar, waveform should be studied more deeply, but it seems more reasonable to provide the system with a first timbre recognition stage, at least at a level of timbral families. Different weight sets could be loaded in the net according to the decision taken by the timbre recognition algorithm before starting the transcription.

A number of issues have to be still faced to have a robust and powerful system, like evolving timbres, noise, and volume variations, but the promising results presented are encouraging enough to keep on researching in this technique.

Acknowledgements

This work has been funded by the Spanish CICYT project TIRIG; code TIC2003-08496-CO4 and Generalitat Valenciana project, code GV04B-541.

References

Boulanger, R., 1999. The CSound Book. MIT Press, Cambridge, Massachusetts.

Brown, J., Zhang, B., 1991. Musical frequency tracking using the methods of conventional and 'narrowed' autocorrelation. J. Acoust. Soc. Amer. 89 (May), 2346–2354.

Hertz, J., Krogh, A., Palmer, R.G., 1991. Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood city, CA.

Hess, W.J., 1983. Algorithms and Devices for Pitch Determination of Speech-Signals. Springer-Verlag, Berlin.

Hush, D.R., Horne, B.G., 1993. Progress in supervised neural networks. IEEE Signal Process. Mag. 1 (10), 8–39.

Klapuri, A., 1998. Automatic transcription of music, M.Sc. Thesis, Tampere Univ. of Technology.

Klapuri, A., 2003. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. IEEE Trans. Speech Audio Process. 11 (6), 804–816.

Lang, K.J., Waibel, A.H., Hinton, G.E., 1990. A time-delay neural network architecture for isolated word recognition. In: Shavlik, J.W., Dietterich, T.G. (Eds.), Readings in Machine Learning. Kaufmann, San Mateo, CA, pp. 150–170.

Marolt, M., 2001. Sonic: Transcription of polyphonic piano music with neural networks. In Proc. Workshop on Current Research Directions in Computer Music, November.

Martin, K., 1996. A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, MIT Media Lab, July.

Moorer, J.A., 1975. On the segmentation and Analysis of Continuous Musical Sound by Digital Computer. Ph.D. thesis, Stanford Univ., Dept. of Music.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.

Shuttleworth, T., Wilson, R.G., 1993. Note recognition in polyphonic music using neural networks. Technical report, Univ. of Warwick, CS-RR-252.

Todd, P.M., Loy, D.G. (Eds.), 1991. Music and Connectionism. MIT Press.

Tolonen, A., Karjalainen, M., 2000. A computationally efficient multipitch analysis model. IEEE Trans. Speech Audio Process. 8 (6), 708–716.

Vercoe, B., 1991. The CSound Reference Manual. MIT Press, Cambridge, MA.