Tatra Mt. Math. Publ. **23** (2001), 141–163

TATRA MOUNTAINS
Mathematical Publications

AUTOMATIC LEARNING COMPOSITION BASED ON POLYPHONIC WAVE SIGNALS AND MUSICAL HISTOGRAMS

Carlos Alberto Lopes da Silva — Luciano Vieira Lima — Wellesley Barros Ferreira — Antonio Claudio Paschoarelli Veiga — Keiji Yamanaka — Yuzo Iano

ABSTRACT. This paper describes the development of an autonomous musical composition system using CLEAN, a functional language based on Lambda Calculus, [D. Cope: *Experiments Musical in Intelligence* (*EMI*) Non-Linear Linguistic-Based Composition Interface **18** (1989), 117–139] and the most popular mathematical tools to analyse signals: the Fast Fourier Transform, Artificial Neural Network, Autoregressive Model and Adaptive Filter (LMS based-procedure algorithms). The system should be capable to create unpublished music with the potential to express feelings similar to those expressed by a model music. It will be possible by frequency identification of musical notes recorded in WAVE format files, and further translation to Standard MIDI Files (S.M.F.). The system analyses the generated MIDI file and identifies its musical harmony. A textual score containing this information is generated. In order to compose new musical scores, the system extracts statistical histograms of notes, rhythm and harmony from these textual scores. The histograms contain the necessary information to generate new melodies for either soloist or polyphonic instruments. The paper also shows that it is possible to design such systems without using the traditional rules or requiring a deep theoretical musical knowledge. Moreover, there is no need of user's intervention or continuous backtracking during the development of new melodies.

1. Mathematical tools

We will begin by approaching the main mathematical tools used in the materialization of the proposed system.

2000 Mathematics Subject Classification: 11K70, 97B20.
Keywords: automated music composition, CLEAN, functional language, wave signal, MIDI, fast Fourier transform.

1.1 Lambda calculus

Programming routines used in the implementation of the developed mathematical algorithms were built in CLEAN, a programming language based on the Lambda Calculus.

The computer Science started with Lambda Calculus. Alonzo Church [Church41] made it clear what a step means. He started by defining the expressions of his system:

1. An identifier, like x, y, z.
2. $(\lambda x \to E)$, where x is an identifier and E is an expression.
3. $(E1\, E2)$, where $E1$ and $E2$ are expressions.

A step is given by any of the rules that transform one expression into another. There are two kinds of the rules:

$\alpha\big(\lambda x \to M = \lambda y[y/x]M\big)$, provided that y does not occur in M.

$\beta\big((\lambda x \to M)N = [N/x]M\big)$.

Church's Lambda Calculus is a formal system, i.e., it is a system that has the following features:

1. An alphabet.
2. A recursive grammar, i.e., a set of rules that can say whether a sequence of alphabetic symbols is an expression in a finite number of steps.
3. A set of inference rules. In Church's Calculus, the rules are α and β. In fact, there is a third rule, but we are not going to use it.
4. Deduction is a sequence $[f_0, f_1, f_2, \ldots, f_n]$, where $f_i \leq n$ is either an axiom or can be obtained by applying the rules to $[f_0, \ldots, f_k]$, $k \leq i$.

1.2 Fast Fourier transform

Let us see how to write algorithms in CLEAN using Lambda Calculus. The reader who is not familiar with this kind of notation should refer to the classical book of Bird and Wadler [Lima98]. One of the most interesting numerical methods is the Fast Fourier Transform described by Jean Baptist Fourier.

According to Heideman [Heideman84], Gauss discovered a method for speeding up its calculation. And, almost 200 years later, Cooley and Tukey rediscovered Gauss' Fast Fourier Transform (GFFT). In this section, a recursive version of GFFT is presented.

Given a function $f(t)$, its Fourier transform is:

$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{i^* 2^* \pi^* f^* t} dt .$$

One can recover $h(t)$ from $H(f)$ through the integral below

$$h(f) = \int_{-\infty}^{+\infty} H(f)e^{-i^*2^*\pi^*f^*t}df\,.$$

The Fourier transform of a sampled function is said to be discrete and is given by

$$H_n = \sum_{k=0}^{N-1} e^{i^*2^*\pi^*k^*n}\,,$$

where $n = 0, 1, 2, \ldots, N-1$. Note that there are n values of H_n, and to calculate each value, it is necessary to perform N multiplications of h_k by the exponential factor. Thus, the number of multiplications is N^2.

Many people consider the Fast Fourier Transform very difficult to understand and implement. It takes about 200 lines in cryptic C code. In CLEAN, the code becomes so clear that any one can understand it:

```
fft [x] = [x]
fft s = front (1.0, 0.0) (fft evens) (fft odds) []
where
    split [] = ([],[])
    split [x0, x1:r] = ([x0:r0], [x1:r1])
    where
        (r0, r1) = split r
    (evens, odds) = split s;
    nr=Polar 1.0 (Pi / (toReal ((length s)/2)))
    front r [] [] r = reverse r
    front r [e:fte] [o:fto] r =
        [e+r*o: front (r*nr) fte fto [e-r*o:r]]
```

The factor is expressed by the term root which multiplies the odd components in the listing of module fft. C programmers use a complex butterfly algorithm to achieve the same result as mathematicians do with Lambda Calculus. It is easy to see that the butterfly algorithm produces the same result as the Clean Fast Fourier Transform.

1.3 Zermelo—Frankel notation

This notation is a very powerful tool when we want to define functions that will work with lists or vectors. Usually, when we write a computer program in procedural languages, we have to create counters, indexes, intermediate accumulators, and so on, to build lists or vectors that follow specific properties. For these cases, mathematics offers us a powerful and efficient tool to treat such problems: the Zermelo Frankell notation (`http://www.hilt.nl/tutorial.html`) proposed by Ernst Zermelo (a German mathematician), and by Adolf Fraenkel (a logistic Israelite) is efficiently implemented by CLEAN. This tool is very important and useful to generate parsers that make Audio→MIDI and MIDI→text conversions.

Let us see a simple example:

Creating orderlies pairs: **CLEAN→ListTupla2 v w** $= [(\mathbf{x}, \mathbf{y}) \backslash\backslash\, \mathbf{x} \leftarrow \mathbf{v} \,\&\, \mathbf{y} \leftarrow \mathbf{w}]$.

Explanation:

ListTupla2 $v\, w =$ list of pairs (x, y) such that x is in v and y is in w.

Then, ListTupla2 $[1, 2, 3]\ [4, 5, 6] = \big[(1, 4),\ (2, 5),\ (3, 6)\big]$.

1.4 LMS adaptive filter

Fig. 1 shows the block diagram of a conventional LMS (Widrow and Hoff) adaptive filter. The structure is formed by a transversal filter whose coefficients "a_{ni}" are updated by using the least-mean-square algorithm [Widrow71]. The input samples at each tap of the transversal filter are weighted by an appropriate coefficient "a_{ni}". The resulting products are summed to form the output signal y_n.

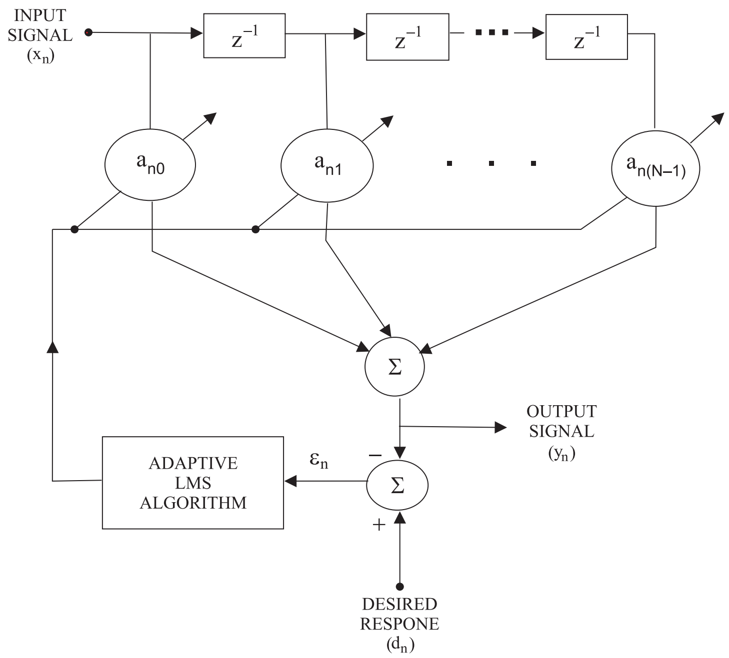


Figure 1. A conventional LMS adaptive filter

Considering the nth time instant, X_n is defined as the input signal vector and A_n as the weight vector whose expressions are respectively:

$$X_n = \left[x_n x_{n-1} \dots x_{n-(N-1)}\right]^T, \tag{1}$$

$$A_n = \left[a_{n0} a_{n1} \dots a_{n(N-1)}\right]^T. \tag{2}$$

The output y_n is given by

$$y_n = A_n^T X_n \tag{3}$$

and the error ε_n by

$$\varepsilon_n = d_n - y_n\,. \tag{4}$$

Through the use of a series of manipulations one arrives at an optimal set of coefficients A_{opt}, which minimizes the mean square error (mse), $E\{\varepsilon_n^2\}$. The optimal solution is known as the Wiener-Hopf equation shown below:

$$A_{\text{opt}} = R^{-1}P, \tag{5}$$

where

$$R = \text{ autocorrelation matrix},$$

$$R = E\{X_n^T X_n\}, \tag{6}$$

$$P = \text{ cross-correlation between } X_n \text{ and } d_n,$$

$$P = E\{X_n d_n\}. \tag{7}$$

The Widrow-Hoff LMS algorithm is a practical method of estimating the values of A_{opt} without matrix inversion given by Eq. (5). It is based on the steepest descent algorithm which continuously updates the weight vector A_n, from sample to sample. The weight update [Malvar92] is defined by the following expression:

$$A_{n+1} = A_n + \mu \cdot X_n \cdot \varepsilon_n\,, \tag{8}$$

where μ is a positive constant that determines the convergence rate and stability.

In order to ensure the stability of the adaptive process, the following condition must be satisfied:

$$0 < \mu < 2/(N \cdot P_{\text{inp}}), \tag{9}$$

where

$$P_{\text{inp}} = \text{ input power},$$

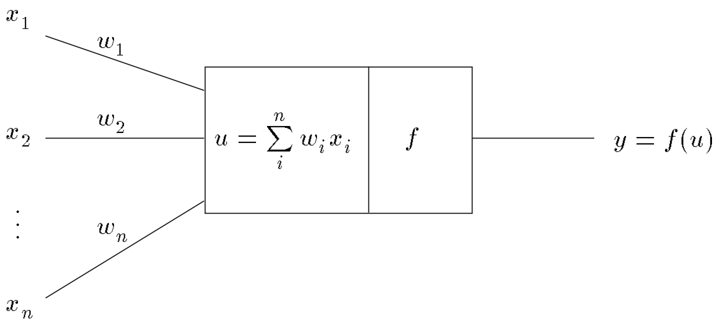
$$P_{\text{inp}} = E\{x_n^2\}. \tag{10}$$

1.5 Artificial neural network

Based on the definition in [Hecht90], a neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and can carry out localized information processing operations) interconnected via unidirectional signal channels called connections. Each processing element has a single output connection that branches into as many collateral connections as desired; each carries the same signal— the processing element output signal. The processing element output signal can be of any mathematical type desired. An artificial neural network can learn to execute a task by examples adjusting its weights based on a training algorithm. The power of ANN has been demonstrated in many applications as in signal processing, control, medicine, business, speech recognition and pattern recognition.

A multilayer perceptron, a type of artificial neural network, trained by backpropagation training algorithm has been used in this work to recognize musical notes.

Artificial neuron model:



where, x_i is the input signal; w_i is the weight of each connection; f is the activation function. The output y is given by the following expression:

$$y = \frac{1}{1 + e^{-u}} .$$

The basic idea used in this system (Wave to MIDI conversion) is to generate a group of coefficients capable to really represent each musical note of the chosen musical instrument. Starting from this point, the classification of these notes is made by using an artificial neural network based classifier. The success of the network will depend on how good those coefficients represent the notes. Our proposal is to model this process using an auto-regressive model of order AR(N). The order of the model is a parameter that depends upon a commitment of the success rate of the classifier and of the number of notes that eventually can be played simultaneously. This method is an alternative to the FFT method when FFT fails.

1.6 Regular expressions, finite automata and context-free grammars

To specify some of possibly infinite languages with finite descriptions of style music notation, regular expression and finite automata are good formal tools. Each regular expression stands for a set of strings: Symbol, Alternation, Concatenation, Epsilon and Repetition (Kleene closure) [Appel98]. With regular expressions we can specify all the symbols used in the text score. With the context-free grammars we can analyse the generated Standard MIDI File and extract the necessary knowledge for set-up the database of musical knowledge.

2. The great problem

Discovering new stars in the sky [Mello98] has everything in common with discovering musical notes in music, in both cases we have continuous signals. In other words, the light of the stars and the sound of a music. The largest faced problem is that it is not known what is being sought. The existent frequencies of the stars are not known a priori as well as the frequencies of the musical notes of the sound signal. The task becomes complicated when we try to apply FFT to the signals. Mello [Mello98] idealized the first mathematical tool to solve the problem of identification of frequencies of stars ignored in a digital map of any section of the sky. When applied to signals whose frequencies are not known, FFT produces, as result, frequencies non-existent in the original signals. Silva et al. [Silv99] sketches a migration of such technology for a possible solution to the problem presented in this paper.

3. The musical notes recognition problem

To be possible to analyse a sound file correctly it is initially necessary to divide into pictures (windows) that contain 2^n points in a signal period, where n is the total number of points of the picture. Such division is necessary to apply Fast Fourier Transform (FFT).

When running FFT with a window of fixed size in a file containing a musical note, the detection of this note is restricted to a maximum size of the window. When analysing a WAVE file containing, for example, only a musical note (C4), we can observe the following facts: when we apply FFT with a picture size with more than 8192 samples, the musical note C4 is found and mistakes are not introduced (presence of other notes). When the size of the picture is reduced (4096–2048–1024, etc.) notes that don't belong to the file start to appear. Starting from a window size of 4096 bytes, it is not possible to identify a musical note accurately. In this case, the ideal window size must be 8192 bytes.

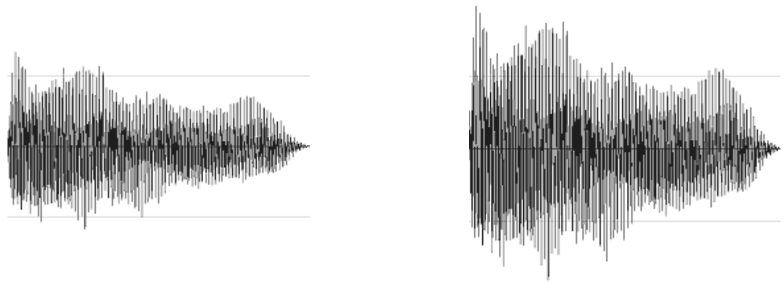
3.1 The musical notes detection algorithm

The algorithm is divided into 3 parts.

1. Normalization of the signal.
2. Forecast of the possible existent musical notes in the WAVE files.
3. Refinement and identification of the existent musical notes.

3.1.1 Normalization

The normalization is done by setting the peak value of the signal to 0 dB. The proposed algorithm is initially not concerned with identification of the volume of the existent musical notes in the files, but has only to guarantee that every existent real note in the file will have a volume close to zero dB. Illustration 1 shows the normalization effect.



(a) (b)

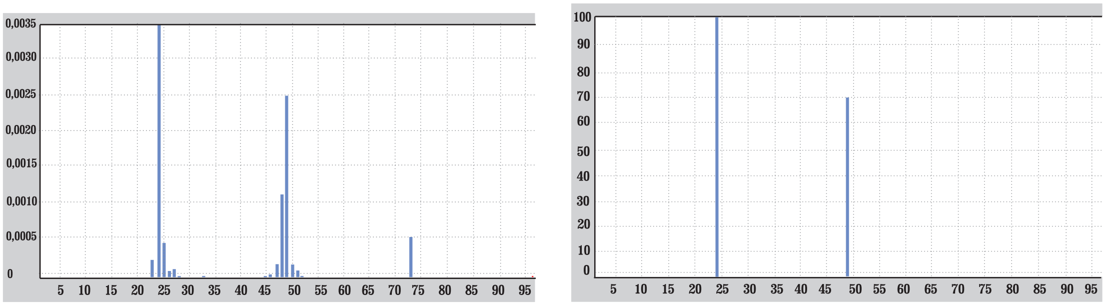
Illustration 1. Waveform of a note C 4. (a) without normalizatin (b) with normalization.

3.1.2 Forecast

In this stage, we will try to foresee which are the possible existent notes. The forecast is made through the analysis of Fourier. With this analysis, the peaks that possess an inferior volume at the defined minimum level for the user are eliminated. We can divide this stage into other two:

a) Filtering of the identified frequencies due to analogy of digital conversion mistake and ignorance of the period whose amplitude has levels inferior dB to the expected ones in the file, since it is normalized to 0 dB.

b) Filtering of the identified frequencies through the note filter of the very temperate scale. In this paper we will assume that the instruments are tuned with the note A (440Hz). Considering the Illustration 2(b), the existent peak in the position 24, is equal to the frequency 258.4 Hz. Consulting a table of tuned musical notes, it is possible to observe that this value approaches to the frequency of 261.63 Hz, that is, to the note C4. The existent peak in the



(a) (b)

ILLUSTRATION 2. Spectrum of Fourier of a note C4. (a) Without Filter. (b) With Filter and normalized from 0 to 100.

position 49 in Illustration 2(a) is equal to the frequency of 527.6 Hz. Consulting the same table of tuned notes, it is possible to observe that it approaches to the frequency of 523.25 Hz, that is, a note C5. Possibly, this second found note should be harmonic from the fundamental.

The expected frequency is obtained through the following formula:

$$F = \frac{(i^* F_s)}{N}, \tag{11}$$

where i is the reference index in the vector of FFT, F_s is the sampling frequency and N is the number of points of FFT. For the mentioned example, the sampling frequency is 44100 Hz, and the size of the sample is 4096.

3.1.3 Refinement and identification

After the forecast of the possible notes, the spectrum obtained is already much clearer, and therefore it just contains possible musical notes. Now, it is necessary to know if the foreseen notes really belong to the analysed file. The main reason of the identification of non-existing notes in the original acoustic sound is that: the Fourier analysis has been applied to a window whose number of points doesn't correspond to the period of the signal producing an aliasing mistake. Another factor is that it didn't have 2^n points in the period analysed by the algorithm of FFT, and in this way, it is not possible an exact identification of the fundamental frequency. This module, refinement and identification, takes the charge of solving this problem.

From the foreseen notes, it is necessary to process them in order to verify their pertinence to the file. That is done by determining the number of necessary points for a period of the signal. This number is calculated by the equation:

$$\mathrm{NP\,1} = \frac{\mathrm{SR\,1}}{\mathrm{F\,1}}, \tag{12}$$

where SR 1 is the sampling frequency, F 1 is the signal frequency and NP 1 is the number of necessary points for this signal. With this value we can calculate

NP 2 with 2^n points that is as close as possible to NP 1. The ideal case is when NP 2 is smaller or equal to NP 1. In this way, the aliasing mistake is avoided. With the value NP 2 and by using the equation (12), a new sampling frequency SR 2 is calculated. Resampling the signal of SR 2 can be done by using the interpolation process. The Illustration 3 shows this process.

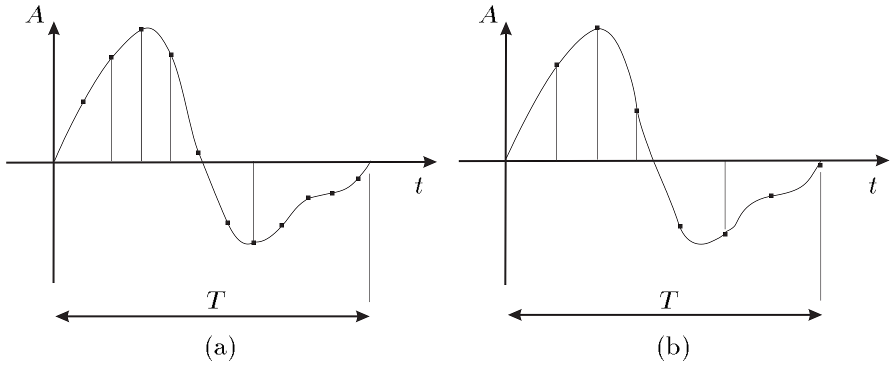


Illustration 3. Resample of an audio signal. (a) Period containing 12 samples (b) Period containing 8 samples.

The Illustration 4 shows how the signal interpolation is done:

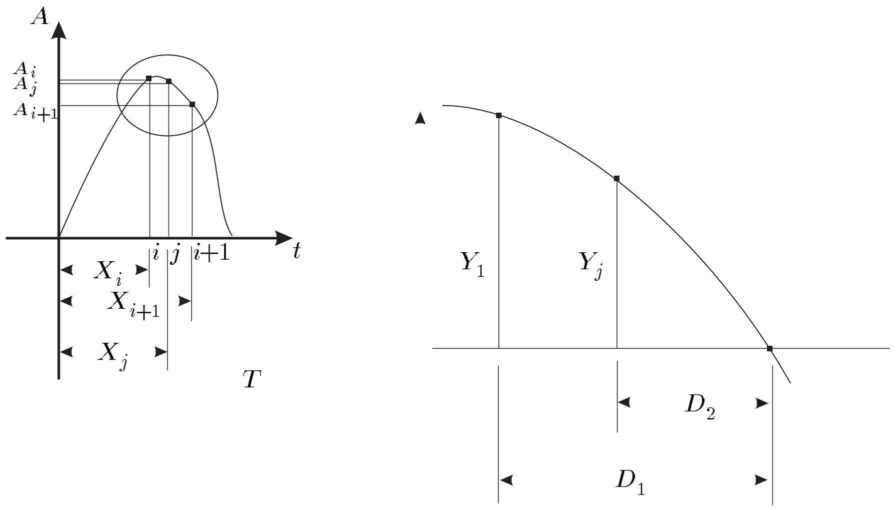


Illustration 4. Interpolating with rates of different sampling.

$$Y_1 = A_1 - A_{1+1}\,, \tag{13}$$

$$X_i = \frac{i}{FS1} e\ X_{i+1} = \frac{i+1}{FS1}\,, \tag{14}$$

$$X_j = \frac{j}{FS2}\,, \tag{15}$$

$$D_1 = \frac{1}{FS1} e\ D_2 = X_{i+1} - X_j = \frac{i+1}{FS1} - \frac{j}{FS2}\,, \tag{16}$$

$$Y_j = (A_i - A^*_{i+1} FS1^* \Big(\frac{i+1}{FS1} - \frac{j}{FS2}\Big)\,, \tag{17}$$

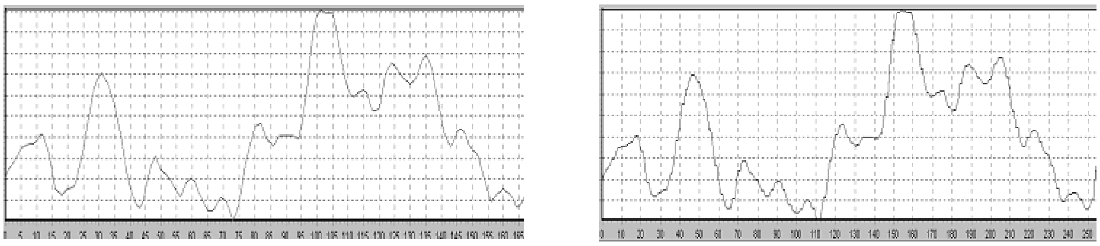
$$A_j = Y_j + A_{i+1}\,, \tag{18}$$

to $1 \leq i \leq FS1$, $1 \leq j \leq FS2$ and $|A_\Phi|\Phi|A_{i+1}|$. To $|A_\Phi|\pi|A_{i+1}$, we have:

$$Y_j = (A_{i+1} - A_i)^*\, FS1^* \Big(\frac{j}{FS2} - \frac{i}{FS1}\Big)\,. \tag{19}$$

$$A_j = Y_j + A_j\,. \tag{20}$$

The Illustration 5 shows an interpolation from 168 to 256 points.

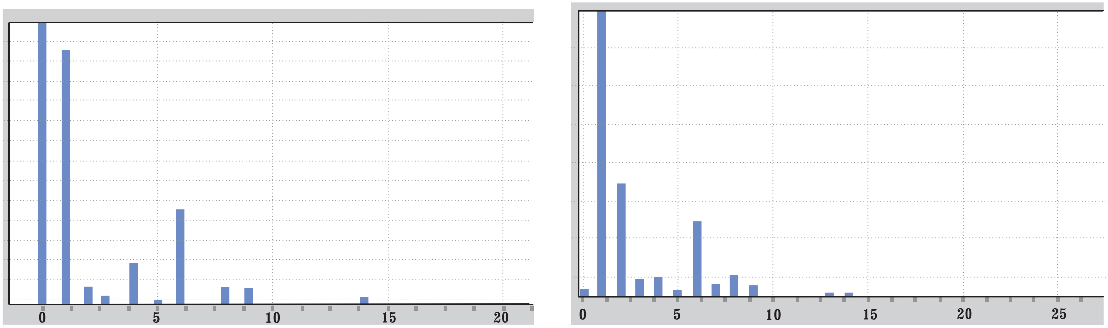


(a) (b)

Illustration 5. (a) The 168 points of a signal C4 that corresponds to a period (b) corresponding interpolation from 168 to 256 points.

After this interpolation, if the frequency really exists, a peak will appear in the position corresponding to the fundamental frequency while FFT is applied, confirming its existence. On the other hand, if we try to find non-existent note, the peak will be zero or very small, as it is shown in Illustration 6. The existence of this frequency must be sought in the whole picture.

Illustration 6(a) shows a set of note frequencies. We previously know that what really exists is the C4 note is confirmed by the algorithm. In other words, we have obtained a maximum peak in the fundamental (index 0). When we try to find a musical note that doesn't exist, its fundamental frequency will not appear in the spectrum. This is shown in the Illustration 6(b). With this process, it is possible to determine the presence of one or more musical notes in an interval



(a) (b)

Illustration 6. (a) Spectrum of Fourier in the search of a note C4.
(b) Spectrum of Fourier in the search of a note D4.

of time exactly specified. Now, it is sufficient to determine the location and the exact duration of these notes.

First, a sweeping in the whole picture looking for the existence of the foreseen notes as well as their beginning and duration, is done. To do this, as a reference it is taken the smallest possible resolution, sweeping the whole picture and applying FFT again. If a frequency or frequencies appear in, at least, two thirds of the minimum resolution, it is considered as a musical note. Then, we can know if the note really exists and where it is located.

3.2 An algorithm for detection of a musical note and its duration

Initially, we should divide the file into a user pre-defined number N of frames.

NA $=$ sample rate of each frame
NR $=$ Increment (one third of the minimum resolution),
$N_R = R_{\min}/3$, where $R_{\min}$ is the size of the necessary sample for the smallest musical note that the user will define.

For good operation of the algorithm, some initial procedures should be taken:

1. Resolution $=$ sixteen notes. A quarter note $= 60/\text{bpm} \rightarrow$ sixteen note $= (60/\text{bpm})/16$.
2. To convert the minimum resolution, previously found, for an amount of samples. Found this amount of samples, we will find the increases NR.

$R_{\min} = t_{R_{\min}}/F_s$, where $t_{R_{\min}}$ is the minimum resolution time and F_s is the sample rate of the WAVE file.

3.3 Time recognition

Shortly, we should analyse the sound file frame by frame in order to detect the existence and the duration of the identified notes. This analysis is done to each third of the chosen minimum resolution. If an identified musical note is found, we must look at the others frames in order to determine when this musical note has stopped sounding. This can be done by analysis of the volume decay of each identified note.

4. Harmony and autonomous composition

Although there are a lot of works dedicated to computer music composition, the majority of them composes on base of strict rules for harmony, stochastic grammar and auto-similarity [Hudak96]. A few of them also uses composition driven by style, where the works of a particular author are searched for the smallest form of repetition, or grains [Smoliar90], to describe style. David Cope [Cope88], [Cope89], [Cope92] describes a system capable to undertaking those tasks. In order to adjust how precisely the system must follow the grammar, to produce the results the user considers adequate, it requires dynamic interaction. This paper describes a new method that can be used to design a system similar to Cope's but without *backtracking and trial and error* procedures.

5. Characteristics of the proposed musical composition system

The aim of this research is to design an autonomous musical composition system that, unlike many others [Cope88] [Cope89], [Cope92], [Dodge85], [Hudak96], [Todd89], presents the following features:

1. It is capable of generating the new compositions without user interaction.
2. It is capable of expressing the feelings, to a listener, similar to those expressed by the model music.
3. It does not need to use explicit rules for harmony and musical composition.
4. It can learn and create music using only the original scores of a general composer and without any previous analysis.
5. It does not use trial-and-error procedure such as that used by M. Gogins [Gogins91] and Cope [Cope89], [Cope92], [Cope88], and also, it does not use backtracking to generate correct notes.
6. If the music generated by the system is fed-back to the system itself, this will not alter its histograms of rhythm, note, and harmony.

5.1 The composition model

Musical composition can be informally classified into two main groups: popular and classical. They can be distinguished, in a more macroscopic level, by the fact that the semantics and the structure of classical music are very specific and explicit for each composition. The model described in this paper was not designed to capture and generate new structures. Once the structure is defined, the system can model and imitate the method used by the author of the original composition to interconnect and generate rhythm, harmony and the notes of a musical score. Therefore, the system is the sole responsible for the composition and connection of the grains that will originate the new music. The generation of those grains should always be done after answering the following question:

Using a randomly chosen note, which should be the next note if one takes into account specific rhythm, the harmony and the personal characteristics of a given author?

This paper addresses this problem and shows a method capable to answer this question.

5.2 Modeling emotions

As described earlier, our intention is to design a system capable of composing music that can express feelings to humans, similar to those expressed by a music composed by a human. If we look around, we will always be able to find maestros with a very deep musical knowledge, sometimes comparable to great classical composers of the past. However, the majority of them, extremely talented maestros, seems not to be able to create music with the same level of harmony and pleasant feelings. This means that a great composer must have more than a deep knowledge of the techniques required for composition and harmony. One can only infer that intuition and social, emotional, and cultural aspects play a very strong role in the process of creation. Hence, since every composer is permanently subjected to unpredictable changes in his life, they should have a considerable effect in his musical style.

5.3 The system

The system has been designed to extract the histograms of rhythm, harmony and melody from a score. Prior to any composition, the original music must be translated into a proper text format (used internally by the system). The new music will be created and stored in Standard MIDI File (SMF) format 1 [Lima99].

An example of a score generated by the harmony computer program with two measures in the format required by the system's parser:

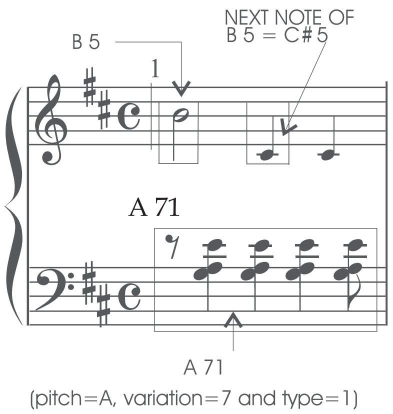
1–(C5 1:4, Eb5 1:4, G5 1:4, D#5 1:4) [C dim 9+ 1];
2–(D5 1:2, f5 1:2) [D m O 1]

The above score follows the grammar shown next.

SCORE $\rightarrow$ MEASURE MOREMEASURE
MEASURE $\rightarrow$ INTEGER "-" (NOTES)[HARMONY]
MOREMEASURE $\rightarrow$ ";"MEASURE
NOTES $\rightarrow$ NOTE MORENOTES
NOTE $\rightarrow$ PITCH RHYTHM
MORENOTES $\rightarrow$ "," NOTES
HARMONY $\rightarrow$ TONE MODE VARIATION TYPE

PITCH $\rightarrow$ (A-G) . ((#)* | (B)*) . (0-9)$^+$
RHYTHM $\rightarrow$ [1-9]$^+$. [0-9]$^+$. (" :").(2^n)
TONE $\rightarrow$ (A-G) . ((#)* | (B)*)
MODE $\rightarrow$ (m | M | dim | + | $-$ | ε)
VARIATION $\rightarrow$ (0-9)$^+$. ("-" | "+" | ε)
TYPE $\rightarrow$ (0-9)$^+$
INTEGER $\rightarrow$ (0-9)$^+$

The next example illustrates the format of a piece of score CPN (**C**ommon **P**ractice **N**otation) for the described syntax.



The transcription of the score (measure 1) to the input format for the system will produce:

1-(B5 1:2, C#5 1:4 , C#5 1:4) [A 7 1],
where
1:2 = half note and **1:4 = quarter note**

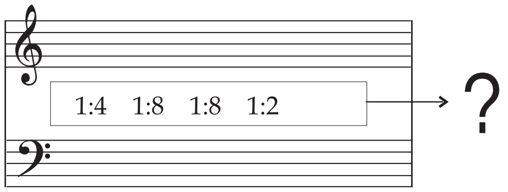
The decomposition of that measure will generate the following three basic components that are also required by the system:

Rhythm = (1:2 1:4 1:4),
Harmony = (A 7 1) and
Melody (solo) = (B5 C#5 C#5)

Based on those information, and using the remaining of the music, the system is capable of extracting the necessary histograms for generating its new compositions. Although, in the above example, it has been used the traditional names for notes and chords, the system can use any chosen name, since, for composition, it does not use any theoretical interpretation. Hence, the B5 note could be called, for instance, note 1, the chord [A 7 1] could be called [chord 1], and so on. The implementation of the note by note generation system, using the concept of next note, is not an easy task, as it may seem at the first glance. A superficial analysis can lead to the conclusion that the only role of the system is to analyse the notes used by the composer in his music and build a data base of the notes that normally follow them. One of the strongest points of the system is that, without using any musical theory, it can avoid cyclic sequences of notes. This is the major differential when comparing the described system to other works using the same basic principles for automatic composition [Cope89], [Cope92], [Cope88]. This interesting feature is due to the fact that we use measure by measure composition. It si possible to do that because every measure of the score must have the same metric. And, therefore, we can use it as the basic cell for the final composition. The distinct strategy used to assembly the histograms, and to create the composer function (to be applied to the histograms), can provide musical structures that, very often, are remarkably similar to those found in the original music.

5.4 The algorithm

The first step towards the design of the system is to create a histogram, which carries the information about every single event of the music, and interconnects them. This is done to retrieve as much information from the music as possible. If this step is done carefully, the histograms should carry enough information to allow the system to create new music, with the potential to express sensations and feelings similar to those expressed by the original music. In order to do so, the system will create three main types of histogram:



RHYTHM HISTOGRAM indicates the rhythm of the next measure, for each measure of the music. This information is based on the statistical probability of the original author choosing the next measures under similar conditions.

A fragment of a typical rhythm histogram is shown as follows:

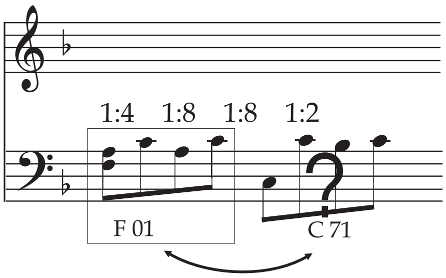
1–<u>Rhythm Histogram</u>:

R1 = (((1:4, 1:4, 1:4, 1:4), Weight = 13/34), Number of Chords = ((1, Weight = 8/13), (2, Weight = 5/13)), (Style = Sonata, Composer = Tchaikowisky, Number of Measures = 34)),
R2 = (((1:4, 1:8, 1:8, 1:2), Weight = 3/34), Number of Chords = ((1, Weight = 1/3), (2, Weight = 2/3)), (Style = Sonata, Composer = Tchaikovsky, Number of Measures = 34)),
RN = ((...), (Style = Sonata, Composer = Tchaikovsky, Number of Measures = 34)).

2–<u>Next Rhythm Histogram</u>:

NextR1 = (((R1, Weight = 9/14,(R2, Weight = 2/14), (R4, Weight = 1/14), (R7, Weight = 1/14), (R12, Weight = 1/14)), (Style = Sonata, Composer = Tchaikovsky, Number of Measures = 34)),
NextR2 = (((R3 Weight = 2/3), (R1 Weight = 1/3)), (Style = Sonata, Composer = Tchaikovsky, Number of Measures = 34)),

NextRN = $\big($(...), (Style = Sonata, Composer = Tchaikovsky, Number of Measures = 34)$\big)$.



CHORDS HISTOGRAM indicates which type of chord can be the next one. Therefore, it is the base for harmony composition. This histogram analyses the rhythm of the current, the previous, and the next measures in a statistical way. This information is based, also, on the statistical probability of the original author choosing the next chord, under similar conditions.

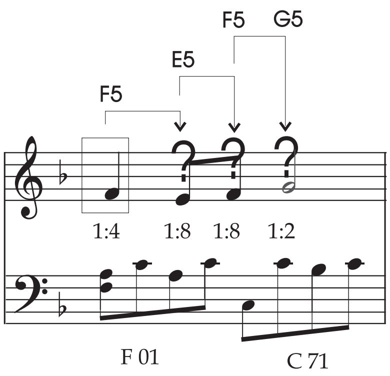
A fragment of a typical chord histogram is shown next.

1–<u>Chord Histogram</u>:

H1 = $\big($((F 7 1), Weight = 5/41, Measure rate = 100 %), (Style = Sonata, Composer = Tchaikowisky)$\big)$,
H2 = $\big($((C 7 1), Weight = 4/41, Measure rate = 100 %), (Style = Sonata, Composer = Tchaikowisky)$\big)$,
HN = $\big($((...)$\big)$, Weight = ..., Measure rate = ... %)$\big)$, (Style = Sonata, Composer = Tchaikowisky)$\big)$.

2–<u>Next Chord Histogram</u>:

NextH1 = $\big($((H2, Weight = 2/5), (H5, Weight = 1/5), (H6, Weight = 1/5), (H26, Weight = 1/5)$\big)$, (Style = Sonata, Composer = Tchaikowisky, Measure rate = 1,0)$\big)$,
NextH2 = $\big($((H1, Weight =2/4), (H3, Weight=2/4)$\big)$, (Style = Sonata, Composer = Tchaikowisky, Measure rate = 1,0)$\big)$,
NextHN = $\big($((...), ..., (...)$\big)$, (Style = Sonata, Composer = Tchaikowisky, Measure rate = ... $\big)\big)$.



NOTE HISTOGRAMS allow the composition of the melody based on the rhythm and harmony generated by the system. The generation of note histogram depends on the rhythm and the harmony of the measure into which each note will be included. The rhythm and the harmony define a semantic environment (structure) into which the note will be included. To clone the feelings expressed by the original music, the note histogram must also be created on the base of the statistical probability of each *next note* being chosen under similar conditions of rhythm and harmony in the original music.

A fragment of a typical note histogram is shown below:

1–<u>Note Histogram for Chords</u>:

Chord H1 = $\big($((G4, Weight = 2/41), (A5, Weight = 13/41), (Bb5, Weight = 2/41), (C5, Weight = 6/41), (D5, Weight = 4/41), (F4, Weight = 5/41), (Pause, Weight = 1/41), (E5, Weight = 1/41), (F5, Weight = 2/41), (A6, Weight = 1/41), (Eb5, Weight = 1/41), (Eb4, Weight = 1/41), (D4, Weight = 1/41), (E4, Weight = 1/41)$\big)$ = 100 %), (Style: Sonata, Composer: Tchaikovsky)$\big)$,

Chord H2 = $\big($((G4, Weight = 2/41), (A5, Weight = 13/41), (Bb5, Weight = 2/41), (C5, Weight = 6/41), (D5, Weight = 4/41), (F4, Weight = 5/41), (Pause, Weight = 1/41), (E5, Weight = 1/41), (F5, Weight = 2/41), (A6, Weight = 1/41), (Eb5, Weight = 1/41), (Eb4, Weight = 1/41), (D4, Weight = 1/41), (E4, Weight = 1/41)?$\big)$ = 100?%), (Style: Sonata, Composer: Tchaikovsky) ?$\big)$.

2–<u>Next Note Histogram for Chords</u>:

NextG4 = $\big($(**Current chord = H1, Next Chord = H1**), $\big($(F4, Weight =1/2), (A5, Weight =1/2)$\big)$, (Style: Sonata, Composer: Tchaikovsky) $\big)$,
NextG4 = $\big($(**Current chord = H1, Next Chord = H2**), $\big($(D5, Weight =1)$\big)$, (Style: Sonata, Composer: Tchaikovsky)$\big)$.
...

5.5 The algorithm for composition

Once the histograms are created, it is almost natural to divide the algorithm for composition into three distinct but strongly interconnected phases: rhythm, harmony and melody composition.

5.5.1 The rhythm composition

The following steps describe the rhythm composition in a simplified way:

1. Choose the desired music or musical style (rhythm) of a particular composer.
2. Define the number of measures to the new music.
3. Apply the rhythm composition function to the **Rhythm Histogram** and define the first type of measure of the new music by a Poisson Distribution [Goodman65].
4. Apply the rhythm composition function to the **Next Rhythm Histogram** and define the next rhythm to the first one [Goodman65].
5. Repeat step 4 until reaching the number of measures defined in step 2.

An example using the three measures defined earlier:

1. (**1:2, 1:2**),
2. (**1:2**) (**1:2**),
3. (**1:2, 1:2**),

The above syntax indicates that measure 1 will have only one chord, i.e., only one set of the brackets in the measure, and two half notes (1:2). Measure 2 will have two chords, i.e., two sets of brackets in the measure, and two half notes. Finally, the third measure will have one chord and two half notes.

5.5.2 The harmony composition

1. Apply the harmony composition function to the **Chord Histogram** and define the first chord of the first measure.
2. Apply the harmony composition function to the **Next Chord Histogram** and define the next chord for the first one [Goodman65]. If the chord uses the whole measure, define the chord of the next measure.
3. Repeat step 2 until every chord of every measure is created. Using the previous example:

$$\begin{aligned}&\mathbf{1}.\ (1:2,\ 1:2)\ [\mathbf{D}+\mathbf{0}\ \mathbf{2}]\,,\\&\mathbf{2}.(1:2)\ [\mathbf{G}-\mathbf{6}\ \mathbf{1}]\,,\\&\quad(1:2)\ [\mathbf{A}+\mathbf{7}\ \mathbf{3}]\,,\\&\mathbf{3}.(1:2,\ 1:2)\ [\mathbf{D}+\mathbf{0}\ \mathbf{2}]\,.\end{aligned}$$

In the expanded example, measure 1 has one chord $[\mathbf{D}+\mathbf{0}\ \mathbf{2}]$ that will be responsible for the generation of two notes. The second measure has two chords $[\mathbf{G}-\mathbf{6}\ \mathbf{1}]$ and $[\mathbf{A}+\mathbf{7}\ \mathbf{3}]$ that should be responsible for the generation of each one note. Besides, the third measure has one chord $[\mathbf{D}+\mathbf{0}\ \mathbf{2}]$ responsible for the generation of two notes.

5.5.3 The melody composition

1. Apply the melody composition function to the **Note Histograms for Chords**, and define the first note of the first measure.
2. Apply the melody composition function to the **Next Note Histogram for Chords** and define the next note [Goodman65] (regardless whether it is a transition between two chords or it belongs to the same chord).
3. Repeat step 2 for every chord of the music until every single one is defined.

   Conclusion of the initial example:

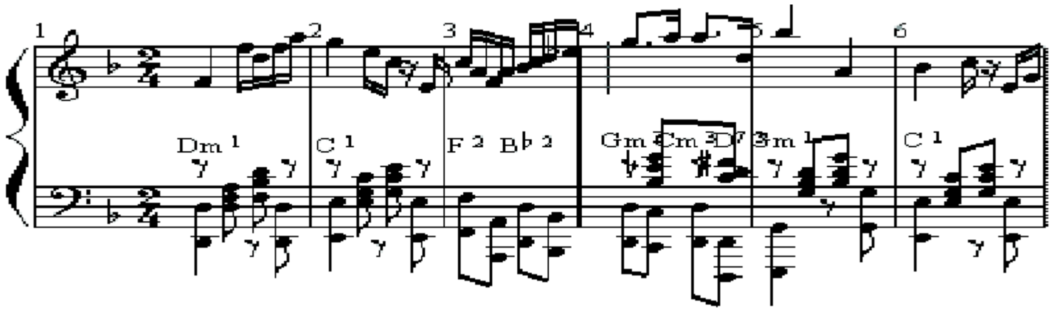
1. (**F#4** 1:2, **F#4** 1:2) [D + 0 2],
2. (**A**4 1:4, **Bb4** 1:4) [G - 6 1],
   (**A**4 1:4, **E**4 1:4) [A + 7 3],
3. (**E**4 1:2, **D**4 1:2) [D + 0 2].

The resulting SMF MIDI file produces the following CPN:

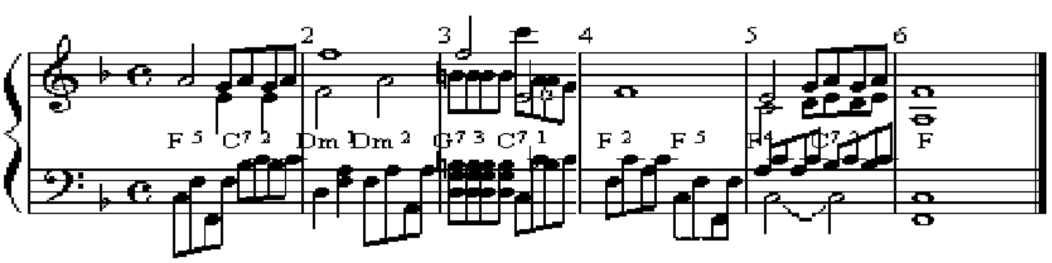
The following examples show fragments of music created by the system based on classical composers and using simplified scores for piano:

1. Six measures generated based on the simplified music for piano None but a Lonely Heart, from P. Tchaikowsky—converted from SMF to CPN.

2. Six measures generated based on the simplified music for piano Cantata Symphony 156, from J. S. Bach—converted from SMF to CPN.



3. Six measures generated based on the simplified music for piano Tristesse, from F. Chopin — converted from SMF to CPN.



6. Some further questions and conclusions

If the system does not add anything new to the histograms of rhythm, chords and melody of a particular author, does it mean that it will not be able to create new music?

No, that is not true. One should not confuse the creation of new music in a certain existent style with the generation of new music in a new style. This system has not been developed to add or impose modifications to the original histograms, but to produce new music capable of inducing feelings similar to those induced by the original music. This feature can be better understood if we look at a very strong feature of Cope's work, to wit, his system recognizes different musical intervals, such as **A-C-E** and **E-C-A**, as a similar and viable description of the style of a particular author. Of course, this is true only in the case of using a specific interval only once. We believe that, if a particular author restricts himself to one of these two intervals, one can considerably modify the style by using the other.

Starting with a chosen structure, how can the system compose new music?

The musician defines the desired structure, and the system will be responsible for composing the motives (melody with rhythm and harmony). For instance, suppose a musician who wants to compose music similar to that created by Wagner. Using the classical example, he may desire to express that love brings both happiness and anxiety, but regardless of the level of anxiety, love can overcome it and make us happy once again. Therefore, the music may induce feelings of happiness to a listener for a few measures. After that, it should induce sadness for a few measures and, eventually, at the conclusion, it must be able to express both feelings at the same time (that could be done by the generation of two voices, each of the inducing an appropriate feeling).

In order to allow this sort of flexibility, the system must store the histograms extracted from different music that indisputably induce those feelings.

7. Conclusion

This paper describes a model approach for music composition that resulted in a system capable of learning and creating new music, without the requirement of procedures such as *trial and error and backtracking*. The system creates new music after the analysis of the scores in the model music obtained from WAVE files and musical analyses. Furthermore, it does so without using any traditional rule for harmony and composition.

Another very important difference between this system and the others based on note by note composition, is the fact that it uses the concept of *measure by measure* composition and due to the fact of extracting the information automatically from the recorded sound file. The *measure by measure* composition uses the harmony (chords) as the central point to decide for next notes and takes into account the statistical weight of the rhythm, chord, and notes histograms in the original melody.

REFERENCES

[Cope89] COPE, D.: *Experiments Musical in Intelligence (EMI)*, Non-linear Linguistic-Based Composition Interface **18** (1989), 117–139.

[Cope92] COPE, D.: *Computer Modelling of Musical Intelligence in EMI*, Computer Music J. **16** (1992), 69–83.

[Cope88] COPE, D.: *Music & Lisp. OH Expert*, March, 1988, pp. 26–33.

[Dodge85] DODGE, C.—JESSE, T. A.: *Composition With Computers — Computer Music — Synthesis*, in: Compositions and Performance, NY, Schirmer, London, Coller MacMilan, 1985, pp. 265–321.

[Gogins91] GOGINS, M.: *Interacted Functions Systems Music*, Computer Music J. **15**, Spring, 1991, pp. 40–48.

[Hudak96] HUDAK, P. ETAL.: *Haskore Music Notation-An Algebra of Music*, J. Functional Language **6** (1996), 465–483.

[Lima98] LIMA, L. V.: *Um Sistema de Composição Musical Dirigido por Estilo*, Tese de Doutorado, Universidade de São Paulo, São Paulo—Brasil, 1998.

[Smoliar90] SMOLIAR, S. W.: *Book Review and Response — Models of Musical Communication and Cognition*, Interface, Vol. 18, 1990, pp. 361–371.

[Todd89] TODD, P.: *Connectionist approach to algorithmic composition*, Computer Music J. **13** (1989), 27–43.

[Appel98] APPEL, W.: *Modern Compiler Implementation in ML*, Cambridge University Press, USA, 1998, pp. 14–75.

[Mello98] MELLO, S. F.: *Estimation of Periods from Unequally Spaced Observations*, Astronomical J. **86** (1998).

[Silv99] *SILVA, C. S. A. L. et al. Use of Fast Wavelet And Fourier Transforms In Music Recognition*, Diderot Forum on Mathematics and Music. 3rd and 4th December, 1999.

[Lima99] LIMA, L. V. ETAL.: *Editing Standard* MIDI *File in Functional Language* CLEAN, Diderot99, Vienna, 1999.

[Goodman65] GOODMAN, R.: *Estatistica*, Editora Pioneira, São Paulo, Brazil, 1965, pp. 65–78.

[Church41] CHURCH, A.: *The Calculi of Lambda-Conversion*, Ann. of Math. Studies, no. 6, Princepton University Press, 1941.

[Heideman84] HEIDEMAN, M. T. ETAL.: *Gauss and the history of the* FFT, in: IEEE Acoustics, Speech, and Signal Processing Magazine, Vol. 1, October, 1984.

[Widrow71] WIDROW, B.: *Adaptive Filters. Aspects of Network and System Theory* (R. Kalman and N. DeClaris, eds.), Holt, Rinehart, and Winston, New York, 1971, pp. 563–587.

[Malvar92] MALVAR, H. S.: *Signal Processing with Lapped Transforms*, Artech House, Boston, 1992.

[Hecht90] HECHT-NIELSEN, R.: *Neurocomputing*, Addison-Wesley, Tokyo, 1990.

Received October 22, 2000

Carlos Alberto Lopes da Silva
Federal University of Uberlândia
BRAZIL
E-mail: carlossilva@quantica.com.br

Luciano Vieira Lima
Federal University of Uberlândia
BRRAZIL
E-mail: vieira@ufu.br

Wellesley Barros Ferreira
Federal University of Uberlândia
BRAZIL
E-mail: bomond@triang.com.br

Antonio Claudio Paschoarelli Veiga
Federal University of Uberlândia
BRAZIL
E-mail: acpveiga@ufu.br

Keiji Yamanaka
Federal University of Uberlândia
BRAZIL
E-mail: keiji@ufu.br

Yuzo Iano
State University of Campinas
BRAZIL
E-mail: rhodan@quantica.com.br