Acknowledgments

The development of the system described in this paper was initiated as a result of a residence program between CRCA (University of California, San Diego), CCRMA (Stanford University) and LIPM, sponsored by the Rockefeller Foundation.

I wish to thank Tim Labor, who patiently tutored my first steps in cmusic, and F. Richard Moore, for their clever and opportune advice and encouragement.

On the Improvement of the Real-Time Performance of Two Fundamental Frequency Recognition Algorithms

ANDREW CHOI
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong

Abstract

Many existing fundamental frequency recognition (FFR) algorithms return reliable results when the analysis window is sufficiently wide. In some applications, however, the response time, i.e., the sum of the width of the analysis window and the computation time for the FFR algorithm, must be made as short as possible. This paper studies the effect of window width on the accuracy of two FFR algorithms and describes a new algorithm with improved accuracy for narrow analysis windows. The new algorithm uses dynamic programming to match harmonics to peaks in the constant-Q transform of the signal. A modification to another FFR algorithm that enhances its performance in real time is also considered.

Introduction

A pitched musical sound is composed chiefly of harmonic components whose frequencies are integral multiples of a fundamental frequency. The problem of fundamental frequency recognition (FFR) is encountered in the automatic analysis of these signals, such as in pitch-to-MIDI systems that enable acoustic instruments to be used as controllers of digital synthesizers.

FFR algorithms that operate in the frequency domain perform spectral analysis on the signal by segments and apply a pattern matching technique to the spectrum to determine each segment's fundamental frequency. Amuedo (1985), for example, identifies sinusoidal components in a signal by the peaks in the power spectrum and examines how the hypothesis for each component to be the fundamental frequency is reinforced by the other components. Pearson and Wilson (1990) consider a multiresolution approach for the spectral analysis step. Doval and Rodet (1991a, 1991b) apply a maximum likelihood analysis to determine the fundamental frequency also using peaks in the power spectrum. Brown (1992) computes the cross-correlation of the constant-Q transform of a segment of the signal with a fixed comb pattern. The calculation of the constant-Q transform and a fast algorithm for approximating it are considered in (Brown 1991) and (Brown and Puckette 1992), respectively.

An alternative approach for designing FFR algorithms is based on computing an autocorrelation between the waveform and a delayed version of itself and determining the fundamental frequency by maximizing the degree of their similarity. Ney (1982) uses time-warping to account for small variations in the signal waveform. The estimated period is the amount of shift that results in the best match of a segment of the signal with a future segment. Lane (1990) adapts the center frequency of a bandpass filter to match the fundamental frequency of the signal using a convergence algorithm. Cook et al. (1993) use a least mean square adaptive algorithm to determine the coefficients of a filter that predicts a segment of a signal from an earlier segment. The phase of the filter is computed from these coefficients, which is then used to estimate the period. Another technique, described in (Brown and Puckette 1993), first determines a coarse estimate of the fundamental frequency using a frequency-domain algorithm. The phase change of the component closest in frequency to the coarse estimate between two segments of the signal separated by one sample is then used to estimate the fundamental frequency accurately.

Accuracy is an important measure of performance of an FFR algorithm. In applications where synthesizers with continuous pitch are controlled, the resolution at which the FFR algorithm can distinguish

frequencies is also an important measure. An error of a few percent in frequency is perceivable in many musical sounds. Frequency-domain FFR algorithms have lower recognition resolution inherently since their spectral analysis step subdivides the frequency range into bins. The disadvantage of autocorrelation algorithms, however, is that a good initial estimate of the fundamental frequency is required for them to converge. Hybrid approaches combine the strengths of both types of algorithms (Kuhn 1990; Brown and Puckette 1993).

When these algorithms are applied to real-time pitch-to-MIDI controllers, another important performance measure is the response time of the system, which is equal to the sum of the width of the analysis window and the computation time for the FFR algorithm. Ideally this response time should be so short that it is not perceived by the performer. Economic and engineering constraints have resulted in commercial systems whose response times are over 50 milliseconds for notes with an average pitch, and even longer for low-pitch notes (Cook et al. 1993). This paper studies the effect of window width on the accuracy of the frequency-domain algorithm in (Brown 1992) and the autocorrelation algorithm in (Cook et al. 1993) and describes techniques for improving their accuracy.

The Constant-Q Transform/cross-correlation Algorithm

The FFR algorithm introduced by Brown (1992) operates in two steps: computation of the constant-Q transform of a segment of the signal, and cross-correlation of the constant-Q transform with a fixed comb pattern that has the logarithmic-scale spacing of the harmonics. The constant-Q transform of a sequence $x[i]$ is defined by

$$X[f] = 1/N_f \sum_{i=0}^{N_f-1} W(N_f, i)\, x[i]\, e^{-j2\pi Qi/N_f},$$

where $Q = 1/(\sqrt[d]{2} - 1)$ is the quality factor of the transform, $N_f = S/(2^{f/d} f_{min})$ is the number of samples of the signal that need to be analyzed for frequency bin f, and W is the Hamming window whose width has been adjusted by N_f, given by $W(N_f, i) = \alpha - (1-\alpha)cos(2\pi i/N_f)$, $\alpha = 25/46$. For real-time operation, and for smaller values of f, N_f may in fact be greater than the number of samples in the signal being analyzed. In these cases, $X[f]$ is computed using only the available samples. The values d, the number of bins to subdivide each octave, S, the sampling rate, and f_{min}, the center frequency of the bin with lowest frequency, are parameters of the algorithm.

The constant-Q transform extracts the frequency components of the signal $x[i]$ in logarithmic frequency spacing, where bin f corresponds to the component with center frequency $2^{f/d} f_{min}$. Since the harmonics of a signal with fundamental frequency f_0 have frequencies f_0, $2f_0$, $3f_0$, ..., and so on, the spacing between harmonics in the constant-Q transform $X[f]$ is fixed and independent of the value of f_0. Thus to correlate the harmonics, a cross-correlation (i.e., convolution) is computed between $X[f]$ and the pattern

$$\overbrace{1,\underbrace{0,\ldots,0,}_{d}\overbrace{1,0,\ldots,0,}^{d\log_2 3}\overbrace{1,0,\ldots,0,1,0,\ldots,0,}^{d\log_2 4}\ldots}^{}$$

$$1,\underbrace{0,\ldots,0,}_{d}1,0,\ldots,0,1,0,\ldots,0,1,0,\ \ldots\ .$$

The number of harmonics to use in the pattern is also a parameter of the algorithm. The center frequency of the bin with the highest cross-correlation value is then returned by the algorithm as the estimate of the fundamental frequency of the signal. To illustrate this algorithm, the constant-Q transform of a 30-millisecond initial segment of a C4 (261.6Hz) note sampled from an electric guitar, the comb pattern, and the cross-correlation are shown in figure 1.

A set of experiments was conducted to study the real-time performance of this algorithm. We implemented and tested this algorithm with the 10-, 15-, 20-, and 30-millisecond initial segments of a set of notes sampled from an electric guitar taken from the range G2 (98.0 Hz) to C6 (1046.5 Hz). The results are shown in table 1. The algorithm correctly recognizes the 30-millisecond initial segments of all notes and fails to recognize most of the initial segments of notes at or below C3 which have length 20 milliseconds or below. It also fails for 10-millisecond initial segments of notes at or below C4. This

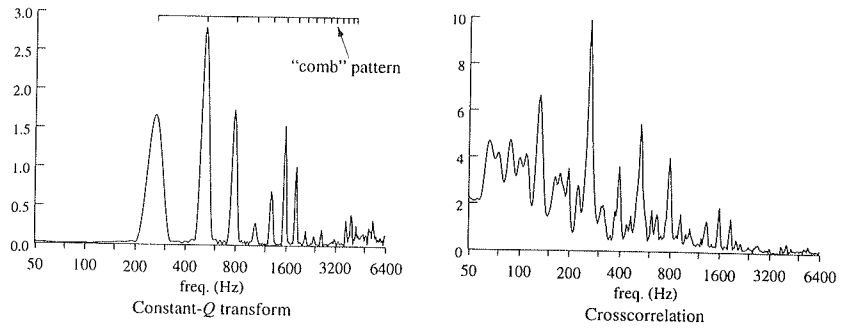

Figure 1: Constant-Q transform and cross-correlation for a 30ms segment of a C4 note.

	G2 (98.0)	C3 (130.8)	G3 (196.0)	C4 (261.6)	G4 (392.0)	C5 (523.3)	G5 (784.0)	C6 (1046.5)
10ms	**50.0**	**59.5**	**80.5**	**105.9**	400.0	526.3	788.5	1052.6
15ms	101.5	**53.7**	197.1	263.1	394.3	526.3	788.5	1052.6
20ms	**50.0**	**52.2**	197.1	263.1	394.3	526.3	788.5	1052.6
30ms	100.0	131.6	197.1	263.1	394.3	526.3	788.5	1052.6

Table 1: Results of constant-Q transform/cross-correlation algorithm (actual frequencies (in Hz) are shown in parentheses beneath note names; incorrect results are shown in bold type).

failure is a result of the inability of the constant-Q transform to distinguish neighboring frequencies when the analyzed signal is short and has a low fundamental frequency. The frequency contents of a bin spill over into neighboring bins causing the cross-correlation step to fail. Such a situation is shown in figure 2 for a 10-millisecond initial segment of a C4 note.

A New Constant-Q Transform/Dynamic Programming Algorithm

The new algorithm is motivated by noticing that although peaks in the constant-Q transforms of problematic cases have broader side lobes, their relative positions remain quite stable. This suggests that higher accuracy can be achieved by replacing the cross-correlation stage by a peak detector followed by an algorithm that matches the peaks to harmonics. In this sense, the new algorithm is a generalization of the one in (Amuedo 1985). Since some detected peaks may be extraneous and peaks corresponding to some harmonics may be missing, a "time-warping" algorithm is devised to match the peaks to the harmonics in a manner that minimizes a total error measure.

Peaks are first identified in the constant-Q transform of the signal segment. To prevent excessive extraneous peaks, ones with small amplitudes are ignored. Examples of such extraneous peaks appear between 600Hz and 700Hz in figure 1. The algorithm identifies and uses the p peaks with the lowest frequencies. Let these peaks have frequencies $f_1, f_2, \ldots, f_p$ and amplitudes $a_1, a_2, \ldots, a_p$, respectively. Also let h be the number of harmonics considered. The values p and h are parameters of the algorithm, chosen to be 10 and 8, respectively, in the experiments described below. Since some peaks as well as some harmonics should be skipped, a matching of peaks to harmonics is represented by a sequence of pairs (i_1, j_1), (i_2, j_2), ..., (i_n, j_n), where $1 \le i_1 < i_2 < \cdots < i_n \le p$, $1 \le j_1 < j_2 < \cdots < j_n \le h$, and $(i_k, j_k) = (i_{k-1}+1, j_{k-1}+1)$, $(i_{k-1}+1, j_{k-1}+2)$, or $(i_{k-1}+2, j_{k-1}+1)$. The last constraint ensures that only a single peak or harmonic is skipped at a time. The boundary conditions are $(i_1, j_1) = (1,1)$, $(1,2)$, or $(2,1)$, and $(i_n, j_n) = (p, h)$, $(p-1, h)$, or $(p, h-1)$. The problem is then one of finding, among all possible such sequences of pairs, a sequence of pairs that minimizes the error measure $E = \sum_{k=1}^{n} e(i_k, j_k)$, where $e(i_k, j_k)$ is the error of matching the i_k-th peak to the j_k-th harmonic.

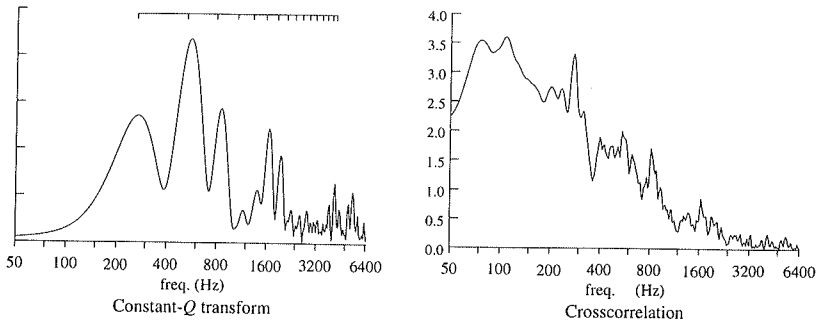

Figure 2: Constant-Q transform and cross-correlation for a 10ms segment of a C4 note.

To formulate the problem so that it can be solved by dynamic programming, let $e(i_1, j_1) = 0$ and let $e(i_k, j_k)$ depend only on the first k pairs of a sequence, i.e., (i_1, j_1), (i_2, j_2), ..., (i_k, j_k). After some experimentation, we arrive at the following definition of the error function e. The matching of the i_l-th peak to the j_l-th harmonic, represented by the pair (i_l, j_l), suggests f_{i_l}/j_l as an estimate of the fundamental frequency. Let $\hat{f}_{k-1}$ be the weighted average of the estimates of the fundamental frequency generated by the first $k-1$ pairs, where the weights are the amplitudes of the corresponding peaks. The assumption is that peaks with larger amplitudes should have a greater effect on the final estimate of the fundamental frequency. Thus define $\hat{f}_{k-1} = \sum_{l=1}^{k-1}(a_{i_l} f_{i_l}/j_l)/\sum_{l=1}^{k-1} a_{i_l}$. The error function is then defined as $e(i_k, j_k) = (f_{i_k}/j_k - \hat{f}_{k-1})^2$.

The recurrence formulas for implementing a dynamic programming algorithm to determine the sequence of pairs with minimum error are derived as follows. Let

- $E_{i,j}$ be the error of a sequence of pairs that optimally matches the first i harmonics to the first j peaks,
- $F_{i,j}$ be the fundamental frequency estimated by this sequence, and
- $A_{i,j}$ be the accumulated sum of amplitudes of peaks used in this sequence.

Then, $E_{i,j}$ is computed by the equation

$$E_{i,j} = \min\{E_{i-1,j-1} + (f_i/j - F_{i-1,j-1})^2, E_{i-2,j-1} + (f_i/j - F_{i-2,j-1})^2, E_{i-1,j-2} + (f_i/j - F_{i-1,j-2})^2\}.$$

The equations

$$F_{i,j} = \begin{cases} F_{i-1,j-1}A_{i-1,j-1}/(A_{i-1,j-1} + a_i) + (f_i/j)\, a_i/(A_{i-1,j-1} + a_i) \\ F_{i-2,j-1}A_{i-2,j-1}/(A_{i-2,j-1} + a_i) + (f_i/j)\, a_i/(A_{i-2,j-1} + a_i) \\ F_{i-1,j-2}A_{i-1,j-2}/(A_{i-1,j-2} + a_i) + (f_i/j)\, a_i/(A_{i-1,j-2} + a_i) \end{cases} \quad \text{and}$$

$$A_{i,j} = \begin{cases} A_{i-1,j-1} + a_i \\ A_{i-2,j-1} + a_i \\ A_{i-1,j-2} + a_i \end{cases}$$

are used to update $F_{i,j}$ and $A_{i,j}$, respectively. Whether the first, second, or third expressions in these two equations are used depends on which term within the braces in the equation for $E_{i,j}$ has the minimum value. Initially, let $E_{1,1} = E_{1,2} = E_{2,1} = 0$ and $E_{i,1} = E_{1,j} = \infty$ for $i > 2$ and $j > 2$. The values of the tables for $E_{i,j}$, $F_{i,j}$, and $A_{i,j}$ can be updated in either column order or row order. References to values outside the range of the indices are assumed to return arbitrarily large values. The final estimate of the fundamental frequency returned by the algorithm is $F_{p,h}$, $F_{p-1,h}$, or $F_{p,h-1}$ depending on which of $E_{p,h}$, $E_{p-1,h}$, and $E_{p,h-1}$ has the smallest value.

	G2 (98.0)	C3 (130.8)	G3 (196.0)	C4 (261.6)	G4 (392.0)	C5 (523.3)	G5 (784.0)	C6 (1046.5)
10ms	**207.3**	**243.2**	201.2	269.1	400.6	527.2	793.2	1052.0
15ms	101.5	134.7	199.3	248.8	396.5	527.0	788.1	1052.0
20ms	102.2	133.5	197.2	263.0	395.1	525.5	788.1	1052.6
30ms	100.0	131.9	196.7	263.0	393.8	525.5	788.5	1052.6

Table 2: Results of constant-Q transform/dynamic programming algorithm (actual frequencies in parentheses; incorrect results in bold type).

The new algorithm was tested on the same set of initial segments of notes, and its results are shown in table 2. It performs better than the algorithm based on cross-correlation, and correctly recognizes the 15-, 20-, and 30-millisecond initial segments of all notes. It also correctly recognizes all 10-millisecond initial segments of notes at or above G3.

Real-Time Considerations for the Periodic Predictor Pitch Tracker

Given an initial estimate of the fundamental period, the periodic predictor pitch tracker (PPPT) of Cook et al. (1993) computes a set of predictor coefficients for the signal using an iterative least mean square (LMS) algorithm and uses them to refine the estimate of the fundamental period. Let the signal be given by the sequence $x_0, x_1, \ldots$ and the initial estimate of the fundamental period be P. Let there be $2M+1$ predictor coefficients $c_{-M}, c_{-M+1}, \ldots, c_M$. The predictor predicts the i-th sample from the $2M+1$ samples centered around the $(i-P)$-th sample using the equation $\hat{x}_i = \sum_{j=-M}^{M} c_j x_{i-P+j}$. The error of this prediction is $\epsilon_i = x_i - \hat{x}_i$. For a given signal, an approximation of the set of predictor coefficients that minimize the mean square error over the prediction of the N consecutive sample values $\hat{x}_{M+P}, \hat{x}_{M+P+1}, \ldots, \hat{x}_{M+P+N}$ can be obtained by iterating the LMS update equations

$$c'_j = c_j + \alpha/(2M+1)\bar{x^2} x_{i-P+j}\epsilon_i,$$

for $j = -M, -M+1, \ldots, M$, over the N predictions $i = M+P, M+P+1, \ldots, M+P+N$. The parameter α is any positive number less than 1 and $\bar{x^2} = 1/R\sum_{i=0}^{R-1} x_i^2$, $R = M+N+P+1$, is the signal power. To perform this operation, the length of the signal must be at least R. Having obtained the predictor coefficients, a more accurate fundamental period estimate is given by

$$P' = P(1 - \theta/2\pi),$$

where

$$\theta = \arctan\left(\sum_{j=-M}^{M} c_j \sin(\omega j)/\sum_{j=-M}^{M} c_j \cos(\omega j)\right)$$

and $\omega = 2\pi/P$.

Cook et al. (1993) suggest using the PPPT in real time by supplying samples to it continuously, i.e., in the above formulation, letting $N = \infty$. The fundamental period estimate will then converge to an accurate value a certain time after the beginning of a note. They report the average of this *latency* to be 30.1 milliseconds for notes between F5 (698.5Hz) and G6 (1568.0Hz), which are tested in their experiments. We implemented and tested the PPPT on the same set of initial segments of notes used in the previous experiments. However, it does not converge to accurate frequency estimates during the duration of most segments of notes, especially for short, low-pitch ones. The algorithm is then modified to choose the largest possible value of N for a signal segment of a given length and iterate a number of times over that segment, allowing sufficient time for the predictor coefficients to converge. Table 3 shows the recognition results of the modified algorithm. In these experiments, M is chosen to be 2, and the initial estimate of the fundamental period is taken to be that of a seminote higher than the note to be recognized. This latter assumption can be satisfied if the PPPT is used as a postprocessing step of the dynamic programming FFR algorithm described in the previous section. The algorithm is set to iterate 50 times over a signal segment. The modified algorithm is found to correctly converge to the fundamental frequencies for all initial segments of notes except in two cases. It cannot be used for

	G2 (98.0)	C3 (130.8)	G3 (196.0)	C4 (261.6)	G4 (392.0)	C5 (523.3)	G5 (784.0)	C6 (1046.5)
10ms	–	**139.1**	193.9	258.5	392.0	525.0	787.5	1052.1
15ms	99.0	131.0	195.1	261.2	394.5	524.1	786.4	1052.3
20ms	98.9	131.4	195.1	261.8	393.7	523.5	785.6	1052.3
30ms	98.9	131.4	195.2	261.8	393.7	523.9	785.6	1052.1

Table 3: Results of PPPT algorithm (incorrect results for 10ms C3; not enough samples to run 10ms G2 case).

the 10-millisecond segment of G2 because the fundamental period of that note is 227.0 samples and the number of samples in the segment is 223 (at a sampling rate of 22255Hz). The algorithm also converges to an incorrect fundamental frequency for the 10-millisecond segment of C3. Furthermore, note that the frequency estimates generated by this algorithm are closer to the actual frequencies than the two frequency-domain algorithms.

Summary

This paper reports experiments that show how the accuracy of the FFR described in Brown (1992) is affected by different window widths. It then proposes a new FFR algorithm with higher accuracy for narrow analysis windows. It also describes a modification to the PPPT algorithm of Cook et al. for improved operation in real time.

References

Amuedo, J. (1985). Periodicity estimation by hypothesis-directed search. In *Proc. of ICASSP '85*, (Tampa, Florida, May 1985), 395–398.

Brown, J.C. (1991). Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am. v. 89*, no. 1, (Jan. 1991), pp. 425–434.

Brown, J.C. (1992). Musical fundamental frequency tracking using a pattern recognition method. *J. Acoust. Soc. Am. v. 92*, no. 3, (Sep. 1992), pp. 1394–1402.

Brown, J.C. and Puckette, M.S. (1992). An efficient algorithm for the calculation of a constant Q transform. *J. Acoust. Soc. Am. v. 92*, no. 5, (Nov. 1992), pp. 2698–2701.

Brown, J.C. and Puckette, M.S. (1993). A high resolution fundamental frequency determination based on phase changes of the Fourier transform. *J. Acoust. Soc. Am. v. 94*, no. 2, Pt. 1, (Aug. 1993), pp. 662–667.

Cook, P.R., Morrill, D., and Smith, J.O. (1993). A MIDI control and performance system for brass instruments. In *Proc. of ICMC*, (Tokyo, Japan, 1993), 130–133.

Doval, B. and Rodet, X. (1991a). Fundamental frequency estimation using a new harmonic matching method. In *Proc. of ICMC*, (Montreal, Cananda, 1991), 555–558.

Doval, B. and Rodet, X. (1991b). Estimation of fundamental frequency of musical sound signals. In *Proc. of ICASSP '91*, (Toronto, Canada, May 1991), 3657–3660.

Kuhn, W.B. (1990). A real-time pitch recognition algorithm for music applications. *Computer Music Journal, v. 14*, no. 3, (Fall 1990), pp. 60–71.

Lane, J.E. (1990). Pitch detection using a tunable IIR filter. *Computer Music Journal, v. 14*, no. 3, (Fall 1990), pp. 46–59.

Ney, H. (1982). A time warping approach to fundamental period estimation. *IEEE Trans. on Systems, Man, and Cybernetics, v. SMC-12*, no. 3, (May/June 1982), pp. 383–388.

Pearson, E.R.S. and Wilson, R.G. (1990). Musical event detection from audio signals within a multiresolution framework. In *Proc. of ICMC*, (Glasgow, 1990), 156–158.

A Linguagem *SOM-A* para Síntese Aditiva

ALUIZIO ARCELA
Laboratório de Processamento Espectral
Departamento de Ciência da Computação
Universidade de Brasília
Brasília, DF — CEP 70910-900
e-mail: arcela@lpe1.cic.unb.br

Resumo

Descrição formal da sintaxe, do universo semântico, dos operadores e de alguns aspectos de implementação da linguagem *SOM-A*. Com o interpretador desta linguagem, que é voltada exclusivamente para a síntese aditiva de sinais musicais, executam-se partituras polifônicas associadas a orquestras, para as quais se definem, uma a uma, as componentes espectrais em todos os seus parâmetros, isto é, ordem de freqüência, ângulo inicial de fase, curva de envoltória, e grau de estereofonia.

HISTÓRICO

Um primeiro esboço para a linguagem *SOM-A* surgiu em 1986 a partir de algumas experiências com o programa Music V (Mathews et. al. 1969). Tais experiências giravam em torno da tentativa de se interpretarem composições algorítmicas baseadas nas árvores de tempos (Arcela 1986), as quais, na maioria das vezes por força do modelo, possuíam uma grande quantidade de componentes espectrais, e eram caracterizadas, do ponto de vista da escrita de eventos e mecanismos espectrais, por não haver nelas, ao menos aparentemente, qualquer possibilidade ou indício de serem interpretadas por um processo que não fosse o da síntese aditiva. Além disso, registrava-se nessas composições a exigência de uma certa projeção acústica estereofônica, segundo a qual um subconjunto bem definido do universo de componentes espectrais deveria soar em apenas um dos canais, enquanto as demais componentes soariam no outro canal, exigência esta que trazia uma ligeira alteração na arquitetura do instrumento mínimo necessário à síntese aditiva padrão, isto é, a que se registra na literatura, como em (Moorer 1977). As estruturas de dados estáticas da implementação FORTRAN do sistema Music V permitiam apenas a execução de instrumentos possuídores de relativamente poucas componentes. E como havia a intenção de se perfazer uma interpretação plena dessas estruturas musicais em todas as suas partes, sem que houvesse qualquer truncamento no conjunto de componentes, a idéia que restava era investir na definição de uma linguagem que pudesse aceitar um instrumento de qualquer tamanho, e que tivesse uma única especialidade: a síntese aditiva. E assim, com apenas 5 operadores, e com uma sintaxe semelhante a de LISP — os instrumentos e os eventos espectrais devem ser escritos rigorosamente na forma de expressões simbólicas —, surgiu a linguagem *SOM-A*, assumindo a simplicidade como sua característica maior.

Como não podia ser de outra forma, a primeira implementação ocorreu em LISP (Nogueira Filho 1988), que era sem dúvida a atitude mais correta e natural, levando-se em conta a natureza das estruturas de dados escolhidas para representar os elementos de *SOM-A*. Naquela época, o que de melhor havia para o sistema operacional MSDOS era o interpretador muLISP87 (Mulisp 1987), um ambiente satisfatório em muitos aspectos, mas oferecendo como obstáculo crucial à implementação de programas voltados para a síntese de sinais de áudio justamente a sua inevitável execução de operações aritméticas totalmente por software. Às vezes, para se interpretar um trecho de 10s, dependendo da complexidade orquestral, eram gastas entre 10 e 20 horas de processamento em intel286 a 10 MHz. Há o caso de uma peça de 1'20" contendo instrumentos de até 76 componentes espectrais para a qual foram necessários 25 dias ininterruptos de processamento! Mesmo assim, a implementação de *SOM-A* em LISP para pequenas máquinas assumiu uma importância capital no desenvolvimento da linguagem, pois, se não consegue executar peças de conteúdo espectral denso em um prazo razoável, ela vem cumprindo o papel de uma especificação