LOW-LATENCY CONVOLUTION FOR REAL-TIME APPLICATIONS

CHRISTIAN MÜLLER-TOMFELDE

ZKM | Institute for Music and Acoustics, Karlsruhe, Germany
now at
GMD - Integrated Publication and Information System Institute, Darmstadt, Germany
mueller-tomfelde@darmstadt.gmd.de

The real-time convolution of a sound signal with measured or computed room-impulse responses poses severe problems as regards computation load, system latency and memory requirements. Some key features of frequency-domain convolution allow to solve the dilemma between latency and computational complexity. Extrapolation from nowadays implementations on off-the-shelf computers allows to sketch future software solutions. In addition the properties in structure and dynamic of the room-impulse responses match ideally with those of the low-latency convolution algorithm for real-time spatial sound processing.

INTRODUCTION

The convolution of a sound signal with the finite impulse response (FIR) of a linear time-invariant system is one of the basic digital sound processing operations and has many applications. In the case of virtual 3D-sound, auralization or reverberation, the convolution can be used to add detailed spatial information to an anechoic sound signal. For real-time interactive environments the latency and the dynamic of the signal processing system are the most evident properties.

1. 3D SOUND PROCESSING

The development of spatial sound processing systems was started in different research areas. In the discipline of room acoustics the halls and other architectural spaces were acoustically measured and their characteristics determined [1]. Then sound path rendering algorithm were developed in order to predict the acoustical properties of rooms before they are realized [2]. Different techniques of auralization are proposed in several publications [3, 4, 5] to playback anechoic sound in modeled rooms. On the other hand, in the area of electronic music feedback delay networks (FDN) were developed to add artificial reverberation to sounds in real time [6, 7]. Digital artificial reverberators were built with computers, synthesizing a reverb with various techniques [8, 3]. Approaches can be roughly identified either as a category of synthesis or a category of simulation. The synthesis technique of reverberation is more perceptive orientated by imitating the acoustical room characteristics with efficient algorithms [9]. Whereas the background of the simulation is a more scientific one and the focus is more on physical propagation models. Nowadays, in the context of multi-media applications 3D or virtual sound processing environments needs to be interactive to give the listener an enhanced convincing sound-room illusion. Like in visual 3D environments the situation must be processed in real-time to reflect the movement of the listener and the changes in the virtual scene around him [10].

1.1. Room Impulse Response

A fundamental description of a reverberant space or room is the so called room impulse response. An enclosed room with reflecting surfaces responds to an impulse excitation with a reverb due to the reflection, the scattering and the diffraction of sound pressure waves. In the signal processing theory the room impulse response (RIR) belongs to the category of FIR filters. The samples of the RIR can be seen as a set of coefficients h_i of a filter h. The filter operation of a signal $x(n)$ with the coefficients h_i is called convolution and can be described as:

$$y(n) = \sum_{i=0}^{N-1} h_i \, x(n-i) \qquad (1)$$

To qualify a reverberant space, e.g., a concert

hall, a church or a living room the RIR normally will be measured with maximum-length sequences or time-delay spectrometry [11]. The responses are preprocessed in the way that various parameters can be extracted from the sound pressure energy distribution in frequency bands, like reverb time RT60 or the early decay time EDT. The general properties of RIR can be used to cluster and extract the information of the characteristics of the measured room. Basic qualities of the room impulse response important in the context of the virtual sound and auralization are:

- Logarithmic decay of the energy
- Discrete early reflections
- Diffuse late reverberation
- Delay until the first wave front
- Rapid and radical changes of the early reflections
- Stable and steady late reverberation

The last two items concern the dynamic properties of a RIR and they can be illustrated by, e.g., the movement of the listening position. The early parts of the impulse response will be changed very quickly while on the other hand the late diffuse reverberations are perceived nearly stable over all positions in a room.

2. BLOCK CONVOLUTION

Due to the long duration the room impulse response (up to $3sec.$ and beyond) it is usually impossible to perform the convolution in the time domain i.e. applying the direct form filter method (see Eq. 1). The transformation of the signal into the frequency domain with the discrete Fourier transformation (DFT) allows to decrease the computational complexity of the filter operation. This is accomplished by the efficient block based algorithm for the DFT called Fast Fourier Transformation (FFT) which reduces dramatically the complexity of the transformation. After transforming a block of N samples and L filter coefficients into the frequency domain the elements of these vectors have to be multiplied. By applying the inverse FFT to the resulting vector the samples of output block will be obtained. This is the complete equivalent operation of the convolution in time-domain (Eq. 1) [12]:

$$x(n) * h(l) \quad \rightleftharpoons \quad X(k)\,H(k) \tag{2}$$

The length of the output block must be equal to that of the direct form filter method in time domain:

$$M = N + L \tag{3}$$

2.1. System Latency

The conversion of the serial sample stream into subsequent blocks of samples is needed to perform the convolution in the frequency domain. This affects the properties of the overall system. Using the described block convolution in a real-time environment, the latency i.e. the time delay from the input to the output is:

$$D = 2\,\frac{N}{f_s}, \tag{4}$$

where f_s is the sampling frequency of the sound signal. Under real-time constrains the algorithm must wait for N samples to have a full input buffer. While waiting for the next N samples the convolution in frequency-domain is done. Then after $2N$ samples the output buffer can be converted to the serial real-time output sample stream. The mechanism works well as long as the computation time of the N-point convolution does not exceed the time span for N samples.

2.2. Computational Cost

The computational complexity of algorithms can be described in numbers of multiplication and addition. The complexity of a real-valued FFT implementation per tab is in general proportional to the logarithm function base 2 plus b constant [12]. As a function of the block size m in the frequency domain (Eq. 3) the complexity can be described as

$$O(m) = a\,\log_2(m) + b \tag{5}$$

By comparison, the computational complexity of the time domain convolution per tab is proportional to the amount L of the filter coefficients. The significant performance improvement of the frequency domain convolution comes along with the linear growing system latency (Eq. 4). For this paper the unit of the complexity is normalized and noted in time per sample.

2.3. Non-Uniform Blocks

Exploration and realization were done to solve the problems of system latency. The theoretical work of Egelmeers and Sommen [13] for the practical situation of acoustic echo canceling and the implementation of Gardner [14] for real-time applications show the possible solution for this dilemma. In both approaches a non-uniform partitioned block-processing algorithm is suggested to combine the advantage of the FFT of the convolution with low-latency res. zero delay time of the signal processing system. Basically the proposed algorithms are splitting the vector

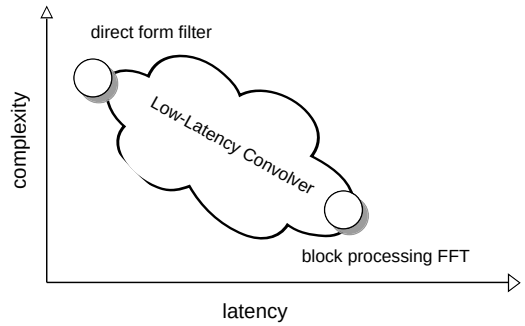

Figure 1: The low-latency convolution is the solution for the dilemma of latency and complexity

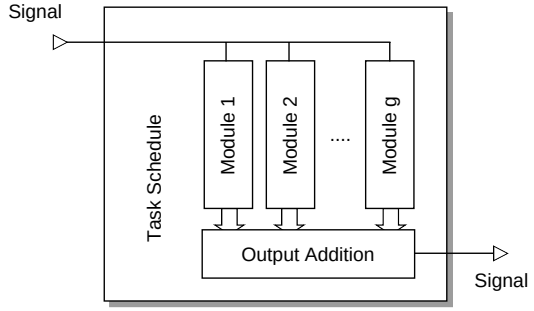

Figure 2: Diagram of the components of a low-latency convolver

of the filter coefficient into blocks of different length. By dividing the convolution problem into several sub problems with different complexity and latency an appropriate mechanism and structure was found to achieve the desired properties [14]. The complexity of the low-latency convolver can be adjusted by a desired input/output delay and the filter length (see figure 1).

2.4. Parallel Filter Tasks

The algorithm mainly consists of concurrently executed tasks, each performing a frequency domain convolution of a subpart of the filter response. The output block of all sub-convolutions are added up to perfectly reconstruct the output signal. A task scheduler has to guarantee the concurrent execution of each sub-convolution with respect to its priority and has to manage the input and output queue of the signal samples (see figure 2).

2.5. Processing Mechanism

To overcome the problems of system latency a practical decomposition of the filter response was introduced in [14] (see figure 3). The first module performs two convolutions each of N-points of the first

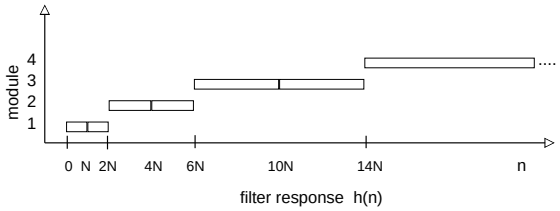

Figure 3: Diagram of the decomposition of an impulse to suffice the real-time constain

$2N$ input samples. The total latency of this module is $4N$ (Eq. 4) and it is enough time for the second module to perform a $2N$-point convolution and to concatenate its output to those of the first module. This mechanism is passed on to the next modules. The segmentation of the filter response and its processing guaranties that no gap occurs in the output. Hence the total latency remains $2N$ taps. The length of the non-uniform block convolution is

$$L(q,g) = \sum_{i=0}^{g-1} 2^{(i+q+1)} \quad , \tag{6}$$

where the q determines the block size 2^q of the first FFT and g the amout of modules. The entire length of the filter is built by the sum over all modules. In cases of desired or accepted delay the latency of the convolver can be adjusted in steps of 2^{q+1} (see Eq. 6).

Additionally in [14] the latency gap of the first modules is filled with the output of a direct convolution implementation to achieve a zero input/output delay and the results of previously transformed signal blocks are used to optimize the performance. Both steps are ignored for this paper. In [13] a more general description of block processing algorithm is introduced, where each module not only performs 2 but rather a variable amount of uniform-partitioned blocks of the filter response.

3. EVALUATION IMPLEMENTATION

In this section an evaluation of the main components for a portable software-based implementation for the low-latency convolution on currently available computer hardware is introduced. In [15] the advantages of portable software signal processing over special-purpose sound hardware are pointed out. The improvement of general-purpose or off-the-shelf computer hardware with recompiled software overtakes the momentary benefits of dedicated sound hardware. A prominent example of this development is the ISPW [16, 17] and the software fts.

To achieve a maximum portability to different computer environments the evaluation programs were written in ANSI-C. Compilers for this level of program language are widely spread and available for nearly all platforms. For the evaluation program different software memory concepts for the loop and pointer variables of FFT implementation are tested with their performance over the length of the convolution. Calculations for the computational complexity based on the measured performances of the components, are extrapolated to upcoming hardware. In the core the implementation of the FFT algorithm follows some recommendations of [18] concerning the variable width, indexing and loop programming. A good compromise in speed and precision is a signal sample representation in a 32 bit floating-point. Almost all modern compilers have built-in optimization for addressing arrays and incrementing loop variables. So the compiler can speedup standard loops and array indexing better than 'hand-optimized' code using cryptic pointer arithmetic.

3.1. Memory Mode: Global, Local or Object

To investigate the influence of the memory allocation space on the runtime performance all basic algorithm parts are implemented in a macro package to create source code at the compile time. This procedure allows to look to three different memory strategies for loop, pointer and accumulation variables used within the algorithm (global, local, object). In the global memory mode all variables of the algorithm are created as global variables, accessible from all parts of the program (main memory segment). In the local memory mode all variables are temporary allocated for the use in a subroutine and the memory space is released by leaving the subroutine (stack memory segment). Additionally with the directive 'register' the compiler can be forced to place the local variables in a CPU-register to enhance their accessibility. The object memory mode is a combination of the other two. A defined memory structure contains all the variables for the algorithm, the memory is allocated dynamical in the heap memory. In some computer environments the access to allocated structured memory will be optimized by using the cache memory which is faster than the access to the main memory.

A macro function `FV(ptr,var)` to access variables is used in the current FFT implementation to select the different memory model for the implementation with a simple compiler switch, which does not effect other program parts. To achieve the best performance of the implementation the FFT source code is also created with macro functions at compile-time.

```
   #ifdef FFT_GLOBAL
#define FV(ptr,var) ptr ##_## var
.  .  .  .
#endif

   #ifdef FFT_LOCAL
#define FV(ptr,var) var
.  .  .  .
#endif

   #ifdef FFT_OBJECT
#define FV(ptr,var) ptr ## ->## var
.  .  .  .
#endif
```

The required memory for the signal blocks is dynamically allocated. When the filter length is equal to the signal block size (Eq. 3) the real-time algorithm needs memory space for $8N$ samples. The input and output queue and the filter block. While the input queue is filled up with new samples and the output is read out a fourth buffer block of $2N$ is needed in which the algorithm operates.

3.2. Performance Platforms

The evaluation program mainly consists of a real value frequency domain convolution with a fixed filter length. The preferred hardware platform were Intel Pentium II based computers with Window NT 4.0 as the operating system. This choice is a good compromise of performance and real-time capabilities at reasonable costs [19]. However the evaluation on high performance UNIX machines like SGI will come in the future. The GNU CC compiler was used to create the executables, other compilers like Visual `C++` can be used to investigate the advantages of the different compilers. The performance test includes a block memory copy operation to manage the signal input and output queue (switched memory queue, SMQ). The evaluation does not include additional commands for a concurrent and parallel process scheduling. These are the planned next steps after this evaluation of the main computational load.

3.3. Performance Benchmarks

The result of an evaluation test run is a pair of filter length of a convolution and the measured time per tap. In figure 4 the performance values are plotted over the length of a frequency domain convolution on a logarithmic scaled x axis. All evaluations of the different CPU's start at $N = 128$ i.e. 2^7 and are processed up to a length of $N = 131072$ i.e. 2^{17}. Hence the time of the filter length varies from approx. $2.9msec.$ up to $2.97sec.$ at a sampling rate of $44.1kHz$. In general the shape of the semi logarith-

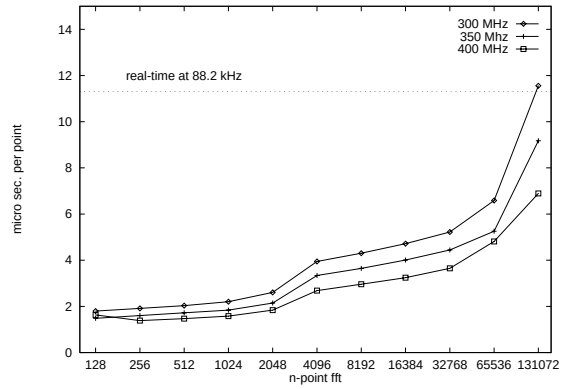

Figure 4: The graph shows the performance of convolver executed with different CPU speeds

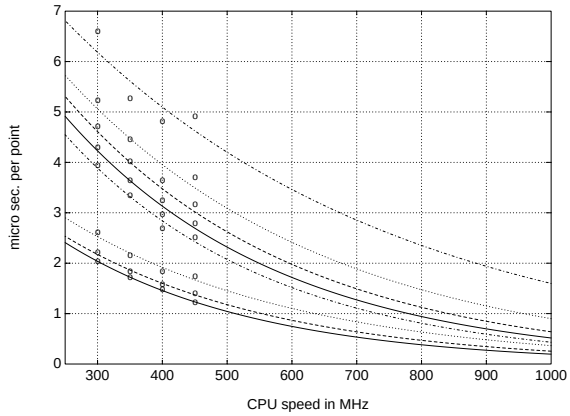

Figure 5: The graph shows the extrapolated performance curves from 2^9 to 2^{16} over existing and expected CPU speeds

mic graphs should be a linear one because of the exponential growth of the taps on the x axis. But the plots of the performances have all nearly the same characteristics, i.e. two steps in the curve, presumably due to the different access speeds to the computer memory (registers, cache and main memory) . The growth of the complexity changes at $N = 2^{12}$ and $N = 2^{17}$, see figure 4.

For each measured performance a regression was calculated to predict the performance values of future CPU's (see figure 5). The two performance steps as mentioned above are represented in this plot as three bundles of curves through extrapolated values.

The array of curves in figure 6 consists on the one hand of a real measured performance on a $400MHz$ CPU and on the other hand of three expected performances for a $550MHz$, $700MHz$ and $1GHz$ CPU for a low-latency convolution with a delay of $2N = 256$.

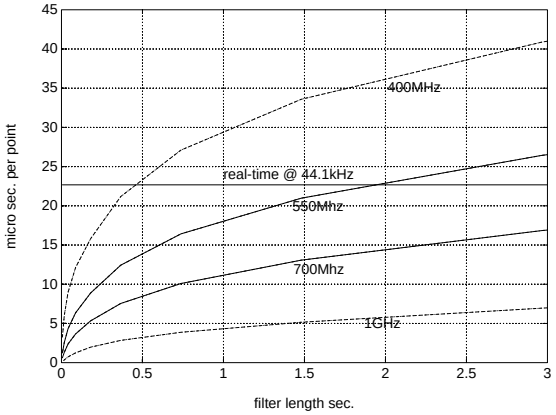

Figure 6: Measured and expected performance of low-latency convolver with different CPU speeds. The system latency is 256 taps

The curves based on the measured and extrapolated data and are calculated with the equation

$$O_{ll}(q,g) = \sum_{i=0}^{g-1} 2\, O_x(2^{i+q}) \quad , \qquad (7)$$

where $O_{ll}(q,g)$ describes the expected performance of low-latency convolution with 2^q as size of the first module and g the numbers of modules. $O_x(m)$ denotes the measured or extrapolated complexity on the CPU system x at filter length m. In figure 6 $q = 7$ which means that latency of these convolvers are 256 taps or $5.8msec.$ at $44.1kHz$.

3.4. Evaluation Results

The extrapolated performance in figure 6 are encouraging. Beyond a CPU speed of approx. $700MHz$ a simple low-latency convolution with a static filter length over $2sec.$ at a sampling rate of $44.1kHz$ seems to be a realistic prediction for the near future. The memory mode tests showed that the use of globally assigned variables has hardly performance advantage over temporary locally created variables whereas the suggested object mode has a poor performance. Therefore all complexity evaluations were done in local memory mode. The extension of the evaluation test program under multi-tasking operating systems to a complete low-latency convolver (as shown in figure 2) will hopefully slightly affect the overall performance. The non-linear performance steps (see figure 4), probably due to memory management and addressing constrains must be considered for future implementation.

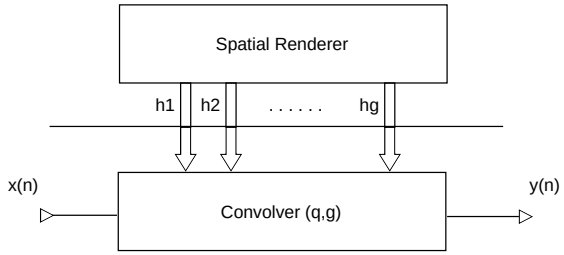

Figure 7: The interface between acoustic render programs and the low-latency convolver

4. IMPULSE RESPONSE INTERFACE

The integration of the low-latency convolution in spatial sound processing environments with an interface is proposed because in the context of virtual 3D sound the convolver meets the requirements of real-time, interactive applications.

4.1. Filter Exchange

The main advantage lies in the similar time-scheme of the non-uniformly partitioned filter response (see figure 3) with those of a typical room-impulse response. The early and quickly changing reflections of the RIR corresponds to short blocks at the beginning of the filter which can be exchanged rapidly. Later more stable parts in the RIR correspond on the other hand to larger blocks with lower update rates.

This scheme can be understood as an interface for the filter coefficients of future real-time rendering programs to the real-time convolver (see figure 7). The update frequency decreases from $345Hz$ at $N = 128$ down to $0.64Hz$ at $N = 2^{16}$ when using a sampling frequency of $44.1kHz$. In a first implementation in [20] the dynamic update of the first 4096 taps of prerendered binaural RIR is performed while the rest of the filter is static. This represents a first and the smallest version of the proposed RIR Interface (RIRI). The influence of the different exchange frequencies within the RIR on the perception has to be investigated carefully.

4.2. Natural Latency

In a virtual spatial sound environment the 'natural' latency i.e. the time of the sound propagation from the source to the receiver can be utilized to reduce the computational complexity of the convolver. A sound source in a great distance allows the convolver more latency as nearer sources hence a real-time control of the latency of the convolver could allow the sound processing environment to save additionally complexity.

5. CONCLUSION

Spatial sound processing in real-time will play an important roll in future multimedia environments for consumer applications like games or entertainment as well as a design tool for engineers and architects. Along with the annually increasing computer performance audio processing software will become powerful on standard PC's. However the latency of a block-based frequency-domain convolution can not be reduced with CPU speed. This can be accomplished by new processing algorithms only. The present work was done during the development of the AML | Architecture Music Laboratory (`http://aura.zkm.de/`), an interactive exhibit of the ZKM |MedienMuseum [21, 22]. The low-latency convolution test programs were intensively used for the evaluation and the selection of RIR before integration within the exhibit [21]. Furthermore, these programs are used by composers who wish to integrate natural sounding reverberation within their computer music compositions [23].

REFERENCES

[1] L. Cremer and H. A. Müller. *Die wissenschaftlichen Grundlagen der Raumakustik, Band I und II.* S. Hirzel Verlag, Stuttgart, 1978.

[2] M. Vorländer. Simulation of transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm. *Journal Acoust. Soc. Amer.*, 86:172–178, July 1989.

[3] M.R. Schröder. Digital Simulation of Sound Transmission in Reverberant Spaces. *Journal Acoust. Soc. Amer.*, 47:424–431, February 1970.

[4] G.S. Kendall and M.D. Moller. Spatial sound effects in a software effects processor. In *ICMC Proceedings*, pages 575–577, Banth, September 1995. Center for Music Technologie, Northwestern University.

[5] K. Kleiner, B.-I. Dalenbäck, and P. Svensson. Auralisation – an Overview. *Journal of Audio Engineering Society*, 41(11):861–875, November 1993.

[6] D. Griesinger. Practical processors and programs for digital reverberation. In *Audio in digital times*, pages 187–195, Toronto, May 1989. Audio Engineering Society.

[7] J.A. Moorer. *About This Reverberation Business*, chapter Perception and Digital Signal Processing, pages 605–639. Foundations of Computer Music. MIT Press, 1987.

[8] J. Stautner and M. Puckette. Designing multi-channel reverberators. *Computer Music J.*, 6(1), 1982.

[9] J.-M. Jot. *Etude et Realisation d'un Spatialisateur de sons par Modèles Physique et Perceptifs*. PhD thesis, Telecom Paris, September 1992.

[10] D. R. Begault. *3D Sound for Virtual Reality and Multimedia*. Academic Press, London, 1994.

[11] Douglas D. Rife. Tranfer-function measurement with maximum-length sequences. *Journal of Audio Engineering Society*, 37(6):419–444, jun 1989.

[12] K.D. Kammeyer and K. Kroschel. *Digitale Signalverarbeitung*. B.G. Teubner, Stuttgart, 1989.

[13] G. P. Egelmeers and P. C. W. Sommen. A new method for efficient convolution in frequency domain by non-uniform partitioning. In *Proceeding EUSIPCO*, volume 2, pages 1030–1033, Edinburgh, September 1994.

[14] W. G. Gardner. Efficient convolution without input–output delay. *Journal of Audio Engineering Society*, 43(3):127–136, March 1995.

[15] Roger B. Dannenberg. A Perspective on Computer Music. *Computer Music Journal*, 20(1):52–56, 1996.

[16] E. Lindeman et al. The architecure of the ircam musical workstation. *Computer Music Journal*, 15(3), 1991.

[17] M. Puckette. FTS: A real-time monitor for multiprocessor music synthesis. *Computer Music Journal*, 15(3), 1991.

[18] A. Freed. Clear, efficient audio signal processing in ansi c. *C Users Journal*, 11(9), September 1993.

[19] R. Dannenberg. The Platform Blues or Looking for Mr. Real Time. *ICMA Array*, 16(1):31–32, 1996. and B. Thom and L. Grubb and Eli Brandt.

[20] B.-I. Dalenbäck and D. McGrath. The narrow gap between virtual realitiy and auralisation. In *Proc. 15th ICA*, volume 2, pages 429–432, July 1995.

[21] P. Dutilleux and C. Müller-Tomfelde. AML, Architecture and Music Laboratory, a Museum Installation. In *Proc. 16th AES Int. Conf. on Spatial Sound Reproduction, Rovaniemi*, 1999.

[22] H. P. Schwarz. *Medien-Kunst-Gschichte*. Prestel, München, Medienmuseum ZKM | Zentrum für Kunst und Medientechnologie, Karlsruhe edition, 1997.

[23] L. Brümmer. Phrenos, ZKM. *CD Magistère et Prix du Concours, "Cultures Electroniques 10"*, page Série Institut/UNESCO/CIME LDC 278 063/64, 1997.