TIME-VARYING FILTER IN NON-UNIFORM BLOCK CONVOLUTION

Christian Müller-Tomfelde

IPSI - Integrated Publication and Information Systems Institute
Fraunhofer - IPSI, Dolivostr. 15, D-64293 Darmstadt, Germany
mueller-tomfelde@ipsi.fhg.de

ABSTRACT

This paper will describe further research on a real-time convolution algorithm for long a FIR filter based on non-uniform bock partitioning. The static behaviour of the algorithm which solves the dilemma between the computational load and the latency of the processing operation is well examined in literature. New directions are investigated to exploit the inherent features of the algorithm and utilise them for audio applications. Especially a dynamic exchange of filter coefficients or subsets of them of a room impulse response is discussed and implemented. Unlike to traditional DSP solutions the prototype is realised in portable software objects and components that can be compiled on multi-propose processing units like off-the-shelf computers with standard audio facilities and different operating systems.

Keywords : convolution, spatial sound processing, real-time, room acoustics, sonification

1. Introduction

In the discipline of three-dimensional audio signal processing many applications demand for massive convolution. In the context of room acoustics and binaural or transaural sound presentation this means that a sound signal is processed by a linear time-invariant system (LTI) with a finite impulse response (FIR) that consists of a large amount of coefficients. The convolution operation can be described by the following well known equation:

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

where x(n) denotes a time discrete series of input samples and y(n) denotes the output. The vector h holds N values which are the coefficients of FIR filter. The equivalence of the convolution operation of the in time-domain (eq. 1) is the multiplication in the frequency-domain :

$$x(n) * h(n) \longleftrightarrow X(k)H(k) \tag{2}$$

This basic signal processing operation (eq.1) requires in a straight forward implementation a lot of computational load while on the other hand an efficient realization in the frequency-domain raises other problems when using it in a real-time environment. The theoretical complexity per sample of eg. 1 is proportional to N, the number of filter coefficients, while in the frequency-domain the complexity can be proportional to the logarithm of N (see [1]). The resulting dilemma of the convolution can be described as the tradeoff between the computational complexity and the latency of the signal processing algorithm. The latency of the algorithm is not directly connected to the available computational power of the target machine: the latency of an implementation will not be reduced by simply employing a faster processing CPU. The algorithm has to be readjusted to the new environment to achieve shorter latencies. This effect is standard to real-time algorithms, where not the frequency of the CPU but sampling frequency of the input sample stream is the time reference for processing. In the context of real-time audio applications a low latency is a desired property because a too long time delay can be identified as an echo by the user, can lead to irritation and the collapse the spatial illusions of virtual acoustical reality applications.

1.1. Related work

The main theoretical work on non-uniform block processing was done by Egelmeers and Sommen (see [2]) for the practical situation of acoustic echo canceling. The implementation of Gardner for real-time applications shows the possible realization in a DSP environment [3]. In addition his realization is a highly efficient solution by using symmetries of the algorithm. Both approaches use the advantage of the Fast Fourier transformation to do an efficient convolution in the frequency-domain. To achieve a low latency resp. zero latency of the signal processing system in both cases the convolution problem is dividing the into several sub problems with a different complexity and latency. A sophisticated organization and parallel operation of the sub problems fulfil required low latency by affordable complexity. In [4] a dedicated hardware environment (see also [5]), that partially correspond to the algorithm was used to build up an example application for real-time virtual acoustics. Finally, possible realizations of the low latency convolution systems on off-the-shelf computer were investigated and their performance extrapolated in [6].
Other approaches for immersive 3D sound scene projection base on feedback delay network (FDN) (see [7][8][9] or [10]). These synthesis techniques for reverberation are more perceptive orientated than the approach of the convolution, they are imitating the acoustical room characteristics with efficient algorithms [11][12].

1.2. Overview

In the following a brief description of the block processing algorithm and the relation to multirate systems is given. Then the system architecture is presented and their parts are described. The processing module which is the core of the architecture is explained in more details. The focus of the next section concerns the dynamic behavior of the system. Mainly the possibility of exchanging the filter coefficients in real-time is considered and strategies are discussed that allow to realize applications for real-time sound projection.

2. Low latency convolution

The algorithm mainly consists of concurrently processing modules, each performing a convolution in the frequency-domain of a subpart of the complete filter response h(n) (see figure 1). The filter response in divided into non-uniform blocks. The output blocks of all sub-convolutions are added up to perfectly reconstruct the response of the overall filter.

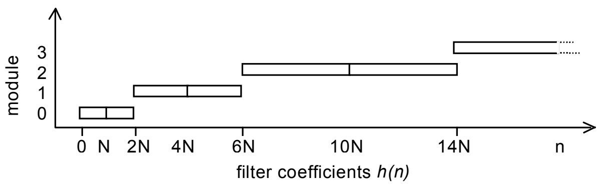


Figure 1. *Diagram of the decomposition of an impulse response h(n) organized in non-uniform blocks*

The input sample stream is converted from serial (sample by sample) to parallel (vectors of samples) to allow the transformation of the signal blocks into the frequency-domain by applying the Fast Fourier transformation. The complexity of the transformation and multiplication is significantly better then the time-domain convolution (see [1],[6]). But with the serial/parallel conversion the block operation obtains the property of latency for real-time applications.

2.1. Processing Mechanism

As shown in figure 2, the processing of blocks of samples in a module has three states over time: *fill* the incoming samples into a block, *process* the block, i.e., applying the filtering in the frequency-domain and finally the *output* of the filter.

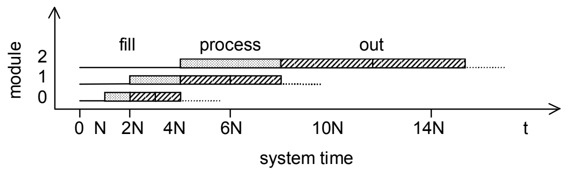


Figure 2. *The different states of the parallel operating modules. The latency of module m is covered by the output of all the preceding modules.*

Under real-time constrains the filling and the following processing of the vector takes a distinct amount of time before the response of the filter appears at the output. This latency can be described by:

$$D_m = 2\frac{N_m}{f_s} \qquad (3)$$

where f_s denotes the sampling frequency and N_m the sample block/vector size of the module m. The decomposition of the filter coefficients as shown in figure 1 has the following effect on the latency of the overall system: The delay of module 0 is 2N so that the first output of the module happens at time 2N and lasts until t=4N (see figure 2). In parallel at t=0 module 1 starts also filling the input signal but the output is delayed by 4N because $N_1 = 2N_0$. Thus the latency of module 1 is exactly the latency plus the output sequence of module 0. In general, the latency of a module m is as long as the latency supplemented by the output of the preceding module m-1. Hence the segmentation of the filter response (see figure 1) and its processing guaranties that no gap occurs in the output of the filter and that the total latency remains $2N_0$ samples. The entire length L of the filter is built by the sum over all modules:

$$L(N_0, M) = \sum_{m=0}^{M-1} 2N_m = \sum_{m=0}^{M-1} N_0 2^{m+1} \qquad (4)$$

Where the M the amount of processing modules and N_0 determines the block size of the first modules m=0. Thus, the number of filter coefficients is not free selectable. These determined lengths of a filter depend also on the latency of the system. In addition, the module processing frequency f_m is defined to be the frequency in which the signal vector of the size N_m is filled:

$$f_m = \frac{f_s}{N_m} \text{ or } T_m = \frac{N_m}{f_s} \qquad (5)$$

Usually the block size of module 0 is in the range 64 up to 1024 samples so that the module processing frequency is in the range of 690 Hz down to 43 Hz (sampling frequency 44100 Hz). Modules with higher order can have processing frequencies of 1 Hz and below.

2.2. Relation to multirate systems

In multirate systems the usable frequency-domain is divided in sub bands. Within these bands filters manipulate input signal and each filter is processed at a reduced sampling rate due to the decimation of the sample stream. A perfect reconstruction of the overall filter output signals is achieved by interpolating and adding up the output of each filter. In the special case of dyadic multirate systems the frequency-domain is non-uniform divided, i.e, the next higher band could have the double width of its predecessor. This structure can be used to represent the critical bands of human audition (for more detail see [13][14]). The latter structure of a of dyadic multirate systems strongly resembles to the decomposition of the filter response in figure 1. Comparing both structures one must swap the meaning of time and frequency. The consequences are briefly discussed without a complete theoretical foundation: The first obvious correspondence is the partitioning of the bands in the frequency-domain of multirate systems with that of the blocks of samples in the time-domain. For multirate systems the block length of N=1 (processing sample by sample) is fixed in every operation within the structure, while for time-domain block processing structures the numbers of bands is always 1, i.e., the full bandwidth is used. The dyadic organization is done in multirate systems among the frequency bands and in block processing systems the lengths of the blocks have a relation based on the factor 2. The correspondence of the signal sampling frequency f_s in dyadic multirate filter band architecture is the block processing frequency f_m in the time-domain.

A detailed description has to be worked out more theoretically. But interesting results could appear from these considerations using a block processing system for audio signal analyze application. Due to the parallel processing of the modules auditory events could be detected in time- and frequency-domain in the modules with different properties: high time precision and low frequency resolution for modules with lower order, i.e., m=0. In modules of higher order the representation becomes more precise in the frequency-domain while the time resolution gets worse.

3. Block processing architecture

The architecture for block processing is simple and well described in literature. In previous work a general theoretical description is provided in [2]. In [4] a dedicated commercial hardware is employed to build up a real-time system with low or zero latency for auralization purposes. The architecture used in this work is designed to be flexible in terms of implementation on standard computer environment and in terms of scalability. The latter property is achieved by using parameterized processing modules that are simple to instantiate and to easy to control in a parallel process environment. Specific optimizations as in [3] or the application of signal processing libraries (like [15]) which make use of processor specific assembler optimization are not considered in the current realization.

3.1. Processing Module

The signal processing inside the module consists of a parenthesis of a Fast Fourier transformation (FFT) operator and its inverse (IFFT) that transform the signal input blocks into the frequency-domain and after a certain processing back into the time-domain (see fig. 3).

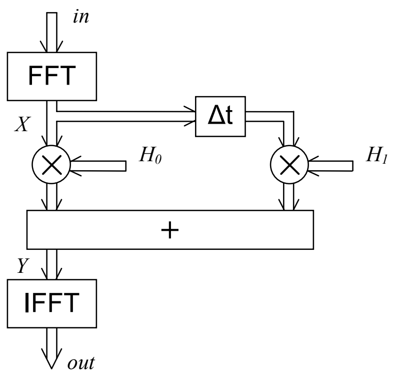


Figure 3. *Signal block processing structure of a module m for frequency-domain filtering. The block size of all the vectors is N_m and the overall latency of this module is $2N_m$*

The operation in the presented module follows mainly equation 2, where the convolution is replace by the multiplication of the corresponding element of the signal vector X and the filter H (Note that the filter vector H is split up into two blocks of the length N_m named H_0 and H_1. The concatenation of H_0 and H_1 is H). The module operation is complemented by a multiplication of previous input signal vector, i.e., the vector X delayed by T_m (see eq. 5), with the filter coefficients H_1. The result of both multiplication are then added and transformed back into the time-domain. The reason for the second multiplication with the filter set H_1 lies in the demand that this module should produce $2N_m$ samples output that fills half of the latency of the next higher module m+1 (see also fig. 1 and 2).

3.2. Processing system

The processing system consists of M modules that operate in parallel. The input signal is converted from serial to parallel, i.e., the samples are grouped into blocks of samples. These blocks are then distributed into the modules depending on the width of their internal block length N_m. A real-time scheduler has to guarantee the concurrent triggering and 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 4).

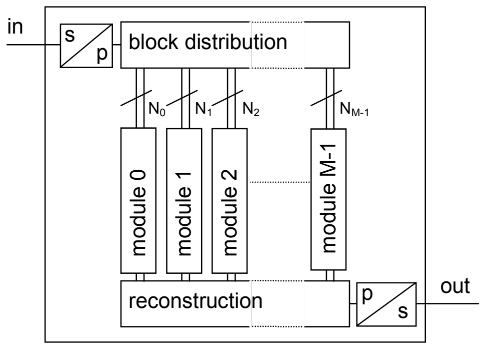


Figure 4. *The architecture of a processing system with the M parallel processing modules, each performing a sub-convolution with $2N_m$ coefficients.*

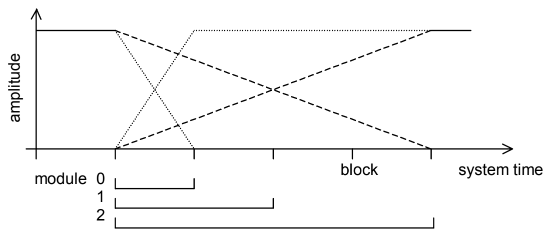
The basic block size N_0 is defined by the first module m=0. It is also the processing base for the distribution and reconstruction unit. The block size of every following module has the double size, so that: $N_{m+1} = 2N_m$. The block distribution unit provides each module m with blocks at the right time, in the right length. On the other side reconstruction unit adds up the output block of all modules to form the response of the overall system, i.e., perfect reconstruction. Thus between the inlet and the outlet the system fulfils eq. 1 with a latency of $2N_0$ samples under real-time constrains.

4. Time-varying filter

For real-time applications in the domain of virtual acoustics not only the change of the input signal with low latency is needed. When the projected scene changes, i.e., the user navigates through the virtual space or simply turns his head also the varying of the filter coefficients has to be considered (as in [16]). Traditionally filter structures operate as static linear time-invariant systems (LTI) where only the input signal varies over time. But this can not be sufficient for a coherent auditory scene illusion where also the scene or, more precise the constellation between sound source and listener is dynamic. Changing all coefficients of a FIR filter (eq. 1) at once can produce sound artifacts or transients within the output signal of the filter. A moderate and secure way to mitigate and avoid these artifact is to use a crossfading unit in the time-domain: to blend from one filter output to the output of the filter with the new coefficients, thus to build a linear time-varying system (LTV). This operation guarantees a smooth transition from one filter output to the new one. The main drawback of such an operation is that two LTI systems need to be computed at the same time. The computational load is increased by the factor 2. In addition, the output of the LTV system is a mixture of two LTI filter signals which may not correspond to a true intermediate out signal. A modification of the module architecture shown in figure 3 allows crossfading with an increased complexity less then factor 2. Therefore the transformation of input vector X is reused for the second temporary FIR filter. The extra computational load vanished after the crossfading operation is finished, i.e., the module falls back into static processing mode. When only a smaller sets of coefficients has to be changed it is useful only partially exchange filter coefficients in the appropriate module.

4.1. Crossfading function

The crossfading process for the initial module m=0 is assumed to range over the complete block size N_0. For the other modules m>0 with greater block sizes different fading strategies can be used. This is useful when concurrent crossfading of multiple modules occurs. The crossfading can either range over the full length N_m of the array (full fade) or always only on the first N_0 elements of the array (primary block fade).



Firgure 5. *For the linear crossfading of the modules different modes are possible: full fade or primary block fade.*

In figure 5 the full fade mode occurs in the doted line for module 0 and in the dashed line for module 2. The primary block fade mode for module 2 is the doted line too. As mentioned above the latter mode concentrates on crossfading only the first block no matter what order of module is actual. This effect is useful when all modules are triggered in that way to crossfade in the same moment and so to achieve a minimum of transition time. The drawback of this mode is discussed in the following.

4.2. Exchange strategies

Depending on the point of time the exchange of coefficients is triggered the delay d_m until the execution of the exchange and the crossfading within a module m reaches the output can range from

$$N_m < d_m < 2N_m \qquad (6)$$

This means that the interference of the trigger time with the block processing frequency can lead to the same time span as the latency of the module. This has certain consequences to the dynamic behavior of the real-time coefficients exchange. The system response time, important for interactive applications is not predictable, only the relation 6 is valid. Thus two strategies are proposed to handle and solve the trade-off between the concurrent exchange and the system response time:

A triggering at the same time, fading at different times
B triggering at different times, fading at the same time

These two strategies of triggering the exchange are illustrated in figure 6a and 6a. Strategy A triggers all modules at once, so that depending of the trigger time a gap can occur in the output. In this gap no output of a crossfading block occurs the reconstructed filter output consists of mixed filter H (see block 5 and 6 after the trigger in figure 6a). The time to exchange a complete set of filter coefficients can last up to twice of the filter length (eg. 4), while the minimum is one filter length. This strategy gives the shortest response time, i.e. maximum $2N_0$. But the behavior of reconstruction of the output is unclear.
The aim of strategy B is to reduce the crossfading time to a minimum of N_0. As show in figure 6b the modules are triggered at different times to guarantee that the output of all modules fade from one filter to the other within the same block of N_0 samples. Therefore the crossfading mode must be set to the primary block mode as described in 4.1. The computational load of each module usually concentrates on the first part of the processing slot, thus the exchange strategy B is fortunately also moderate in terms of additional computational load. The additional filter load is spread over time. The main drawback of the strategy B is that system response time will be d_M (eq. 6), that means that the coefficients can changed at most at the block processing frequency of module M of the system f_M.

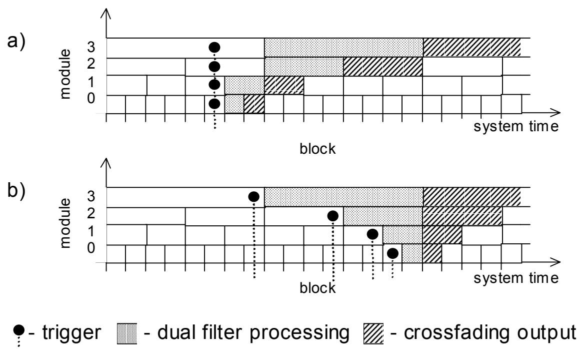


Figure 6. *Different ways to initiate the full exchange of coefficients of an example system that consists of four modules: a) all modules are triggered at the same time b) all output changes at the same time.*

5. Application: Real-time sound projection

The following suggested application should demonstrate the usage of the non-uniform block processing for an individual dynamic sound projection in real-time. The listener is placed in the center of a virtual sound field while the sound source has a fixed orientation and position (see figure 7). For simplification only the horizontal plane is considered. In this example module 0 of the system contains 16 sets of sub filters that correspond to the view angle of the listener. The next module has only 8 distinct set, etc.. When the user now turns his head in the horizontal plane with a constant speed each module has to update the filter coefficients in a different frequency: the lower the order of the module the higher the update frequency, while those of modules with higher is less. The hierarchy of the update frequency matched ideally with the time structure of a room impulse response so that rapid early reflections are considered more often than the late reverb parts. On the other hand audio path rendering algorithm can take advantage of this timing structure of the different modules where the direct path and early reflections can be calculated faster than the late reverb.

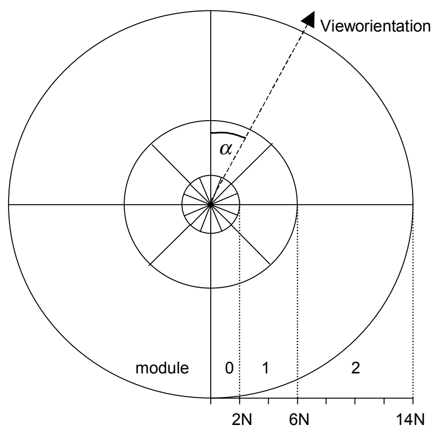


Figure 7. *The arrow and the angle alpha illustrate the head orientation of the listener in the horizontal plane. The circles represent the modules and the segments the time-varying filter coefficients of each module.*

6. Implementation

The current implementation of the convolution prototype is realized with portable software objects and components for multi-purpose CPUs without a graphical user interface. Different areas of computer system programming like multi threading, inter process communication, real time (RT) clock/process and process priority scheduling are involved by the implementation of the described non-uniform block algorithm on a off-the-shelf computer. A general interface guarantees access to these functions under different operating systems (OS). Although these features are nearly state of the art in all operating system, like multi thread processes, some like true real-time process support is not widespread. Currently three OSs are used as a target environment: Window98, BeOS, and OpenStep. The first is used the have a comfortable programming environment, the second was chosen due its RT capabilities and the third is used as a general Unix reference. It is planed to recompile the convolution program also under Window 2000,Unix systems and MacOS X. The file and audio i/o is by now merely implemented to provide the basic functionality for demonstration and validation purposes. The software is strictly written in ANSI-C following guidelines from [17][18][19] concerning the art of signal processing. Despite of using C instead of C++ as the implementation language an object-orientated (OO) like approach of programming is realized to combine speed of execution of direct function calls and flexibility of programming for the implementation. The described modules are realized, e.g. as a software object class which can be instantiated dynamically multiple times during runtime. The Windows implementation suffers under the non optimal real-time properties of the operating system, so that the theoretical load of the hardware resources is hardly reached because of too many drop-outs in the signal output. Therefore BeOS 5 and its RT capabilities were used and the behavior of the convolution system operated as expected. An implementation of the low latency convolution under Max MSP or in a Matlab environment seemed not feasible because of the lack of parallel processing features with different priorities.

7. Conclusion

In this paper described further research on a real-time convolution algorithm based on non-uniform bock partitioning. The test implementation on off-the-shelf computers encouraged the author to adapt the low latency convolution algorithm for the use beyond standard static filter applications. The realisation of a time-varying filter system is demonstrated and the different update rates of the processing system are exploit and used the fit in applications of real-time sound projection.

8. REFERENCES

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

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

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

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

[5] Huron , Lake DSP, http://www.lakedsp.com/

[6] Müller-Tomfelde, C., Low Latency convolution for real time application, In *Proceedings of the AES 16th International Conference*: *Spatial Sound Reproduction*, Rovaniemi, Finland, 1999 April 10-12, pp. 454-460.

[7] Stautner, J. and Puckette, M., Designing multichannel reverberators. Computer Music Journal, 6(1), 1982.

[8] Moorer, J.A., About This Reverberation Business, chapter Perception and Digital Signal Processing, pp 605-639. Foundations of Computer Music. MIT Press, 1987.

[9] Griesinger, D., Practical processors and programs for digital reverberation. In Audio in digital times, pp 187-195, Toronto, May 1989. Audio Engineering Society.

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

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

[12] Dutilleux, P., Müller-Tomfelde, C., AML: Architecture and Music Laboratory, In *Proceedings of the AES 16th International Conference*: *Spatial Sound Reproduction*, Rovaniemi, Finland, 1999 April 10-12, pp. 191-206.

[13] Zölzer, U., Digital Audiosignalverarbeitung Vorlesungsskriptum der TU Hamburg Harburg.

[14] Fliege, N.J., Multirate Digital Signal Processing. Wiley, 1994.

[15] Signal Processing Library, Intel, http://developer.intel.com/software/products/perflib/spl/

[16] Wenzel, E.M., J.D. Miller, and J.S. Abel, "A software-based system for interactive spatial sound synthesis," *ICAD'200*0, May 2000.

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

[18] Dannenberg, R.B., The Platform Blues or Looking for Mr. Real Time. ICMA Array, 16(1):31-32, 1996.

[19] Dannenberg, R.B., A Perspective on Computer Music, Computer Music Journal, 20(1):52{56, 1996.