REAL-TIME PARTITIONED CONVOLUTION FOR AMBIOPHONICS SURROUND SOUND

Anders Torger

University of Parma
Industrial Engineering Dept.
V.Scienze 181/A, 43100 PARMA, ITALY
torger@ludd.luth.se

Angelo Farina

University of Parma
Industrial Engineering Dept.
V.Scienze 181/A, 43100 PARMA, ITALY
farina@pcfarina.eng.unipr.it

ABSTRACT

Ambiophonics, as one of the most realistic music reproduction methods, requires multi-channel convolution with very long impulse responses for creation of believable reverberation.

In this paper it is shown how the heavy processing task required for a real-time Ambiophonics system, or similar convolution-intensive system, can be handled by a low-cost personal computer, by means of partitioned convolution.

The only theoretical advantage of partitioned convolution is that it provides low input/output delay. However, since the intensive part of it is very easily made optimal for the target platform, which normally is not the case for the standard overlap-and-save algorithm, it often provides the fastest convolution as well.

1. INTRODUCTION

The Ambiophonics surround system [1], is based on the coupling of two different reproduction methods: transaural presentation of a cross-talk cancelled stereo recording, and multi-channel 3D reconstruction of the room's reverberation by means of convolution with multiple IRs. Ambiophonics was pioneered by Glasgal [2], employing initially a physical barrier for the cross-talk cancellation, and multiple hardware convolvers for the reverberation.

A first important advancement was the substitution of the physical barrier with a DSP-system capable of signal processing for cross-talk cancellation. This was possible thanks to the stereo dipole concept initially developed by Kirkeby, Nelson and Hamada [3], and further refined with the help of one of the authors [4].

The surround part of an Ambiophonics system was however still based on expensive hardware convolvers, until the first software-based multi-channel convolvers were made available by Lopez [5] and by one of the authors [6].

The goal of this paper is to describe the partitioned convolution algorithm employed inside the multi-channel software convolver in [6], and discussing its performance in comparison with two other arithmetically cheaper algorithms: traditional overlap-and-save, described in the Oppenheim-Shafer book [7], and non-uniform partitioned convolution described by Gardner [8]. It will be shown that although the convolution method employed here is obviously less efficient than the other two, it is much easier to optimize the code at low level for target processors such as Intel Pentium III/4 or AMD K6/Athlon, leading to a CPU load which in many cases is significantly lower.

An additional advantage is that partitioned convolution allows for a small overall latency, becoming a serious contender to the hybrid zero-delay convolution algorithm patented by Lake Technology and clearly described by Gardner, since it has a significantly lower computational load [8], and is trivial to implement. This becomes even more evident in cross-talk cancellation applications like Ambiophonics, where delay is introduced in the filters themselves, thus there is little value in having a zero-delay convolution algorithm. The delay needs only to be low enough so the resulting system can be used interactively.

1.1. Brief description of an Ambiophonics system

The method can be basically explained as the simultaneous superposition of two very different systems: cross-talk cancelled reproduction over a pair of closely-spaced loudspeakers, and approximate wavefront reconstruction with an Ambisonics array, being fed with reconstructed hall ambience signals derived from the left and right direct sound channels convolved with a set of weakly-correlated real hall impulse responses. Figures 1 and 2 show the basic scheme of the two parts of the system.

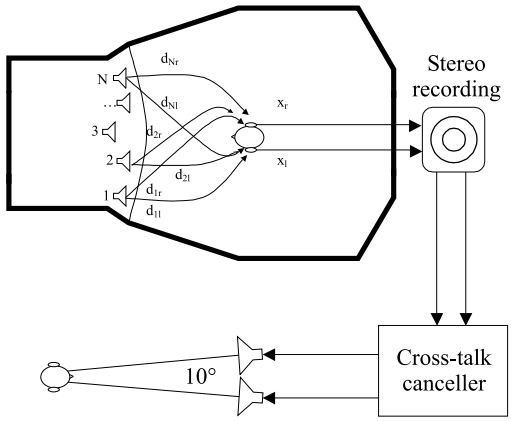

Figure 1: *Stereo-dipole reproduction through cross-talk canceling digital filters.*

The cross-talk cancellation operation is performed through the convolution of the two input signals with a set of four inverse filters. These filters can however be quite short, the major processing demand is instead introduced by the reproduction of off-stage early reflections and reverberation tails, which are reproduced through the surround array consisting of eight or more loudspeakers.

For each surround loudspeaker, two convolutions are necessary, one for the left and and one for the right stereo channel. The IRs employed must be unique, and be near the full length of the

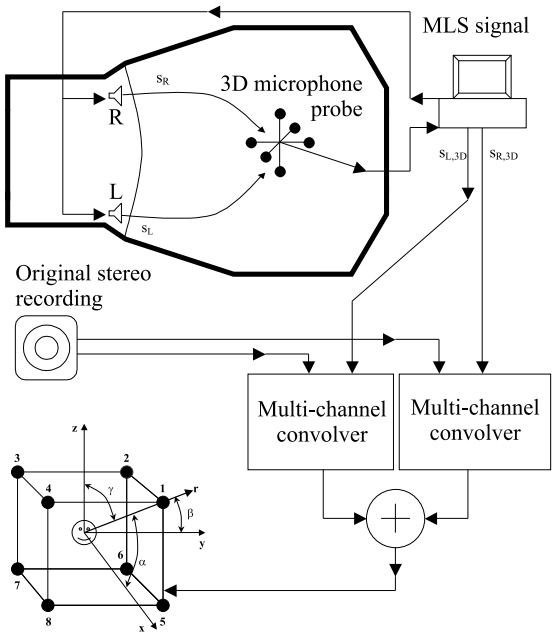

Figure 2: *Virtual Ambisonics reproduction by convolution with two sets of 3D impulse responses.*

reverberation time of the original hall, which in extreme cases can be several seconds.

In [1] both the filter design for the cross-talk cancellation and the aspects of deriving reverberation filters from original hall measurements are thoroughly addressed.

2. DESCRIPTION OF THE ALGORITHM

A brief explanation of the well known frequency-domain convolution algorithm is given here, both in its unpartitioned and partitioned forms.

2.1. The unpartitioned Overlap-and-Save algorithm

Although this frequency-domain convolution algorithm is not the one employed here, it is useful to review it quickly, as it is fundamental to understanding partitioned convolution, which will be described in the next sub-section.

The convolution of a continuous input signal $x(\tau)$ with a linear filter characterized by an impulse response $h(t)$ yields an output signal $y(\tau)$ by the well-known convolution integral seen in equation 1.

$$y(\tau) = x(\tau) \otimes h(t) = \int_0^\infty x(\tau - t) \cdot h(t) \cdot dt \qquad (1)$$

When the input signal and the impulse response are digitally sampled ($\tau = i \cdot \Delta\tau$) and the impulse response has finite length N, such an integral reduces to a sum of products (equation 2).

$$y(i) = \sum_{j=0}^{N-1} x(i-j) \cdot h(j) \qquad (2)$$

The sum of N products must be carried out for each sampled datum, resulting in an enormous number of multiplications and sums! Due to this, the real-time direct convolution is normally limited to impulse response lengths of a few hundred points, while a satisfactory emulation of a typical concert hall requires at least 65,536 points, at 48 kHz sampling rate.

However, the convolution task can be significantly simplified performing FFTs and IFFTs, because the time-domain convolution reduces to simple multiplication in the frequency domain, between the complex Fourier spectra of the input signal and of the impulse response. As the FFT algorithm inherently assumes the analysed segment of signal is periodic, a straight-forward implementation of the frequency domain processing produces unsatisfactory results. The periodicity caused by FFTs must be removed from the output sequence, which can be done with the overlap-and-save algorithm [8]. The flow chart in figure 3 explains the process.

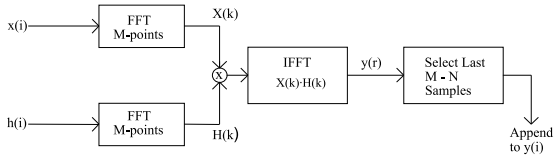

Figure 3: *The Overlap-and-Save algorithm.*

As the process outputs only $M - N$ convolved data points, the input window of M points must be shifted to the right over the input sequence of exactly $M - N$ points, before performing the convolution of the subsequent segment. The trade-off is that FFTs of length $M > N$ are required. Typically, a factor of two ($M = 2 \cdot N$) gives the best efficiency to the overlap-and-save algorithm.

2.2. The partitioned Overlap-and-Save algorithm

In this variation of the basic algorithm, which was first proposed by Stockham in 1966 [9] and further refined for real-time implementation by Soo and Pang in 1986-1990 [10], the impulse response h is initially partitioned in a reasonable number P of equally-sized blocks s_n, as seen in figure 4.

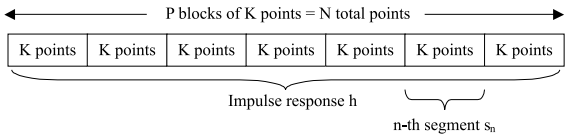

Figure 4: *Impulse response partitioning.*

Each of these blocks is treated as a separate impulse response, and convolved by a standard overlap-and-save process, making use of FFT windows of length L. Each block is zero-padded to the length L (typically equal to $2{\cdot}K$), and transformed with FFT so that a collection of frequency-domain filters s_n is obtained. The results of the multiplications of these filters with the FFTs of the input blocks are summed, producing the same result as the unpartitioned convolution, by means of proper delays applied to the blocks of convolved data. Each block of input data needs to be FFT transformed just once, and thus the number of forward FFTs is minimized.

The main advantage compared to unpartitioned convolution is that the latency of the whole filtering processing is just L points instead of M, and thus the I/O delay is kept to a low value, provided that the impulse response is partitioned in a sensible number of chunks (8 – 32). Figure 5 outlines the whole process.

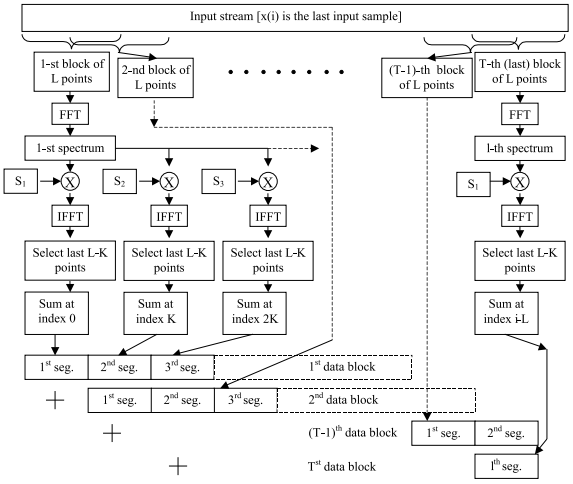

Figure 5: *Partitioned convolution.*

3. COMPUTATIONAL PERFORMANCES

From a theoretical point of view, the process described in the previous sub-section is less efficient than the unpartitioned overlap-and-save algorithm. It requires a higher number of arithmetic operations, and more memory references. It does do some earnings on the FFT calculations, since one M point FFT is replaced with P L-point FFTs, but with the slow increase of computational load of the FFT algorithm as M grows larger ($M \cdot \log(M)$), the performance increase is (theoretically) small.

The cost added by partitioned convolution is the P multiply steps with $P-1$ sums needed for each L-point output block. With a typical value of P = 16, the number of memory references is increased 16 times, the multiply step is performed 16 times more often, and the sums are not even needed in unpartitioned convolution! However, the same memory is referenced several times, and the total amount of memory referenced is slightly reduced, since the input and output buffers are only L points each, instead of M points as for the unpartitioned case. Additionally, the multiplication of spectra and the subsequent sum takes only a fraction of operations compared to FFT, so the cost is not as high as it may seem at first glance.

From a practical point of view, there are some non-obvious advantages of partitioned convolution. First of all, as the size of the FFTs exceeds the cache size of the processor, the performance degradation is larger than theoretically expected. Furthermore, as the number of partitions go up, the major part of the computational load is moved from the FFT calculations, to the simple multiply and add step, which is very easily optimized. On modern processors this translates into a hand-coded SIMD assembler loop of a few lines of code. The SIMD instruction set allows for executing several arithmetic operations in a single instruction, making the resulting code very efficient.

To increase the performance further, this critical loop is extended with cache preload instructions, which improves memory access performance. The task of optimizing the FFT algorithm is much more complex, and therefore the available implementations seldom make the most out of the given hardware. Here, the well-known and highly efficient FFTW library [11] is employed for the FFT calculations.

We use the open-source BruteFIR convolver [8], which was originally designed with Ambiophonics in mind, but has become a general-purpose audio convolver. In this case, it is configured for 10 channel Ambiophonics, meaning 2 inputs and 10 outputs, with 2 IRs per output, a total of 20 independent ones having the lengths indicated. The sampling rate is 48 kHz and the internal resolution is 32 bit floating point. Table 1 and 2 show how large part of the processor time available is needed for achieving real-time operation. Thus, a lower value is better, and a value larger than 1.0 means that the machine cannot handle the processing in real-time.

The performance benchmarks clearly show that the partitioned convolution algorithm actually outperforms unpartitioned convolution in all measured cases, if a suitable number of partitions is chosen. Considering the added benefits of reduced latency and more flexible IR lengths, partitioned convolution should always be preferred. The performance comparison between the two approaches, becomes largely a comparison of the speed of the FFT implementation and the small assembler loop performing the multiply/add step, as can be seen in table 3. FFTW works better on Intel processors than on AMD, thus the gain from partitioning is less evident on the Pentium test system. The higher memory bandwidth of the AMD test system improves the performance of the multiply/add step which is strictly limited by memory bandwidth, like most algorithms on today's standard computers.

Since Ambiophonics uses many more filters than there are inputs and outputs, unpartitioned convolution has an advantage over partitioned. This is due to that partitioned convolution gains speed from moving processing time from FFT to the multiply/add step, and FFT is done per input and output (12 in total), while multiply/add is done per filter (20 in total). Thus, in cases where there is one filter per input and output even larger performance gains by using partitioned convolution should be expected (which actually can be seen in table 4). However, in the rare cases where an optimal FFT algorithm is available, unpartitioned overlap-and-save will out-perform partitioned convolution in terms of throughput, as observed from informal tests made with Lopez software [5]. Lopez uses a heavily optimized proprietary library from Intel, which unfortunately is both closed-source and limited to the Microsoft Windows platform, and is therefore not employed in BruteFIR.

IR length	unpartitioned	4 part.	8 part.	16 part.
32,768	0.59	0.53	0.59	0.78
65,536	0.68	0.61	0.64	0.80
131,072	0.74	0.81	0.73	0.86

Table 1: *Ambiophonics performance on a Pentium III 550 MHz, 256 kB cache, 100 MHz RAM.*

3.1. Comparison with non-uniform partitioning

As described by Gardner [8], partitioned convolution can be arithmetically optimized by using non-uniform partitions, starting with

IR length	unpartitioned	4 part.	8 part.	16 part.
32,768	0.37	0.28	0.32	0.43
65,536	0.56	0.42	0.34	0.45
131,072	0.64	0.51	0.59	0.47

Table 2: *Ambiophonics performance on an Athlon 1000 MHz, 256 kB cache, 266 MHz RAM.*

	I/O	FFT	Mix/scale	Multiply/add
unpartitioned	6 %	71 %	16 %	7 %
partitioned	7 %	15 %	17 %	61 %

Table 3: *CPU time distribution – N = 131,072.*

a small size to provide low I/O delay, and increasing the size further back in the impulse response for increased efficiency. This way a large part of the multiply/add step can be traded for FFTs of longer length. The processing time required grows much slower with increased IR lengths than with uniform-sized partitions, thus it is a more scalable algorithm. However, it is much more complex to implement since it requires scheduling and synchronisation of parallel convolution tasks.

To compare the two partition strategies we have created a dummy program which does not carry out real convolution but performs the FFTs, spectral products and sums required for non-uniform partitioned convolution, and the corresponding for the uniform and unpartitioned case. The program simulates convolution of a single filter with one input and one output. In table 4 the results are presented. The IR lengths have been chosen to be as near as possible to 131,072 while exactly fitting into the non-uniform partitioning scheme with constant processor demand suggested by Gardner [8]. The execution times have been divided with the time for 131,072 taps unpartitioned convolution, thus a value higer than 1.0 means slower than that reference. The lowest latency of 1024 samples was chosen since at 48 kHz it is near the practical limit of a personal computer implementation (about 20 ms). The 1000 MHz Athlon computer from table 2 generated the results.

As seen, the non-uniform partitioning is never faster than the unpartitioned case, but can provide very low latency without a dramatic increase in execution time. However, for Ambiophonics, which is targeted at home use and reproduction of previously made recordings, high throughput is central and latency is of less importance, and therefore uniform partitions is the better choice.

IR length	114,688	122,880	126,976	130,048
latency	16384	8192	4096	1024
exec-time / unpartitioned	0.45 1.06	0.71 1.19	1.33 1.27	5.25 1.45
number of partitions	14 6	30 8	62 10	254 14

Table 4: *Uniform (top) vs. non-uniform (bottom) partitioned convolution.*

4. CONCLUSIONS

The partitioned convolution algorithm presented here makes it possible to employ a standard, low-cost personal computer for implementing a complete Ambiophonics surround sound processor. This gives access to the Ambiophonics technology for a wide number of users. The same processing can also be used for other audio applications which require multi-channel convolution with very long impulse responses, such as multi-channel reverberation, wave front synthesis and detailed equalization of loudspeaker arrays.

The algorithm proved to be very efficient in terms of usage of computer resources. In the currently available public domain implementation, it can provide higher throughput than the traditional unpartitioned overlap-and-save and the non-uniform partitioned convolution algorithms, which both are theoretically cheaper.

5. ACKNOWLEDGEMENTS

This work was supported economically and technically by the Ambiophonics Institute, founded by Ralph Glasgal. The authors express here their gratitude to and admiration for him, for having developed the Ambiophonics technology and having promoted it for mass deployment completely royalty-free and without patents, giving substantial support to the research on surround sound reproduction and advanced digital signal processing.

6. REFERENCES

[1] A. Farina, R. Glasgal, E. Armelloni, A. Torger, “Ambiophonic Principles for the Recording and Reproduction of Surround Sound for Music”, 19th AES Conference, Schloss Elmau, Germany, 21-24 June 2001.

[2] R. Glasgal, K. Yates, “Ambiophonics – Beyond Surround Sound to Virtual Sonic Reality”, Ambiophonics Institute, 1995.

[3] O. Kirkeby, P. A. Nelson, H. Hamada, “The Stereo Dipole – A Virtual Source Imaging System Using Two Closely Spaced Loudspeakers”, J. AES vol. 46, n. 5, 1998 May, pp. 387-395.

[4] O. Kirkeby, P. Rubak, A. Farina, “Analysis of ill-conditioning of multi-channel deconvolution problems”, 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk Mountain House New Paltz, New York October 17-20, 1999.

[5] J. J. Lopez, A. Gonzalez, “PC Based Real-Time Multichannel Convolver for Ambiophonic Reproduction”, 19th AES Conference on Surround Sound, Schloss Elmau, Germany, 21-24 June 2001.

[6] A. Torger, “BruteFIR – an open-source general-purpose audio convolver”, http://www.ludd.luth.se/~torger/brutefir.html

[7] A. V. Oppheneim, R. Schafer, “Digital Signal Processing”, Prentice Hall, Englewood Cliffs, NJ 1975, p. 242.

[8] W. G. Gardner, “Efficient convolution without input-output delay”, J.AES vol. 43, n. 3, 1995 March, pp. 127-136.

[9] T. G. Stockham Jr., “High-speed convolution and correlation”, AFIPS Proc. 1966 Spring Joint Computer Conf., Vol 28, Spartan Books, 1966, pp. 229 - 233.

[10] J. S. Soo, K. K. Pang, “Multidelay block frequency adaptive filter”, IEEE Trans. Acoust. Speech Signal Process., Vol. ASSP-38, No. 2, February 1990.

[11] M. Frigo, S. G. Johnson, “FFTW: An Adaptive Software Architecture for the FFT”, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 3, 1998, pp. 1381-1384.