45º CONGRESO ESPAÑOL DE ACÚSTICA
8º CONGRESO IBÉRICO DE ACÚSTICA
EUROPEAN SYMPOSIUM ON SMART CITIES AND ENVIRONMENTAL ACOUSTICS

EFFICIENT PARAMETRIC HRTF IMPLEMENTATION WITH GPUS

PACS REFERENCE 43.60.Gk

Ramos, Germán[1]; Belloch, Jose Antonio[2];
(1) ITACA Institute, Universitat Politècnica de València
(2) ITEAM Institute, Universitat Politècnica de València
Camino de Vera s/n
Valencia – 46022
Spain
Tel: +34 963879604
e-mail (1): gramosp@eln.upv.es
email (2): jobelrod@iteam.upv.es

ABSTRACT

HRTF (Head Related Transfer Functions) techniques are nowadays used in applications like 3D audio, immersive teleconferencing systems, gaming, etc. They allow helping in creating a more real experience, locating the sounds at the desired positions in space. This paper, presents an efficient parallel implementation of HRTF sets using GPU (Graphic Processing Units), based on a low-order parametric modelling technique of the HRTF developed and adapted by the authors. The proposed approach allows obtaining a low computational cost and low database size HRTF sets, together with the acceleration obtained with the parallel implementation with GPUs. Hence, it is possible to achieve an accurate HRTF model that could be used in different environments with a great number of sound sources in real time.

RESUMEN

El uso de las funciones de transferencia de la cabeza (HRTF, Head Related Transfer Functions) es hoy en día habitual en múltiples aplicaciones como audio 3D, sistemas de teleconferencia inmersiva, videojuegos, etc. Mediante estas técnicas, se obtiene una experiencia acústica más real, permitiendo localizar los sonidos en el espacio. Este artículo presenta una implementación paralela eficiente de HRTFs empleando GPU (Graphic Processing Units), basándose en una evolución de modelos paramétricos de las HRTFs desarrollados y adaptados por los autores. La propuesta permite obtener una implementación de bajo coste computacional y de un menor tamaño de la base de datos de las HRTF, junto a la aceleración del cálculo en paralelo con GPUs. Los resultados demuestran que se pueden obtener modelos realistas de HRTFs que pueden ser usados en distintas aplicaciones, con un gran número de fuentes de sonido en tiempo real.

1. INTRODUCTION

Nowadays, Head-related transfer functions (HRTFs) are used in different applications where realistic audio is needed. HRTF capture all the effects that a free-field sound wave suffers from its source to the listener's ear canal [1]. This includes the acoustic path, the torso and head reflection and diffraction effects, and the outer ear. This information is used by the brain to localize the source in the space, comparing the responses from both ears. The ITD (Inter-aural Time Difference), ILD (Inter-aural Level Difference), shading effect on the frequency response, and reflection patterns due to the torso an external ear are compared [2], [3], as seen at Fig. 1.

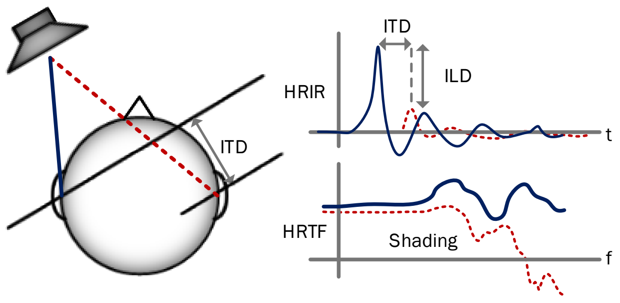

Fig. 1 – HRTFs and HRIR (Head Related Impulse Response) effects

HRTFs are used in immersive sound applications like stereo enhancement and spatial sound reproduction, virtual reality, auralization, videoconference systems, and videogames [4]. In all of these real-time applications, achieving low-order models of the HRTFs is desirable for reducing the computational cost, while trying to keep the perceptual characteristics of the original HRTF.

Different solutions have been proposed for HRTF modeling. Some make a computational approach designing analytical models of the head and simulate the wave propagation and diffraction effects. Others have an empirical approach [3], using digital filters to model the HRTF behavior, as it will be done here. The use of Finite Impulse Response (FIR) filters is straightforward for HRTF modeling by doing the proper time windowing and preprocessing of the HRIR [3], where filter orders close to 200 have been suggested for acceptable results. Infinite Impulse Response (IIR) filters have been also used [3], requiring, in general, lower orders than FIR. Different filter design methods have been used like Prony's method and Yule-Walk, common pole and zero modeling, balanced model truncation, and genetic algorithms [5-8].

Recently, one of the authors proposed at [9] an IIR parametric model implemented directly as a SOS (Second Order Section) chain, where the SOS are forced to be a second order low-shelving and peak filters, all of them defined by its parameters (frequency, gain, and quality factor Q) [10], [11]. The use of parametric methods allows describing the HRTFs with a limited set of parameters, instead of the whole coefficient set (FIR and IIR), reducing the amount of information needed to model the HRTFs, and hence the database size. Other benefit of using a parametric model as in [9] is the simplification of the interpolation procedure of HRTFs responses at positions where they have not been modeled and measured. With a simple interpolation of the parameter's value of the SOS, an interpolated HRTF could be obtained using the parameter values of the neighbor modeled positions. This approach to the interpolation problem is simpler and gives satisfactory results with a much lower computational cost demand as traditional and complex cross-fading techniques in time [12] or frequency domain [13] solutions.

During the last years, the use of GPU (Graphic Processing Units) for accelerating different calculus problems non-related with graphics has been extended and adopted in different fields due to its high computational capacity and internal parallelism (with hundreds or even thousands individual computational units). Actually, GPUs are inside any computer, tablet or smart-phone, being a computational resource available for the developers and programmers not only for graphic purposes. In audio, and particularly, in HRTF modeling, some of the authors have been working on massive multichannel FIR convolvers [14] and on IIR parallel filter banks [15], related with the work proposed here.

This works is focused on getting the benefits of the internal parallelism of GPUs for the parametric modelling of HRTFs. For that, the original parametric model of [9] that is formed as a SOS chain, must be changed and transformed to a parallel one in order to be able to parallelize its calculus. The paper is organized as follows. At section 2, a brief introduction of the original parametric model of HRTFs [9] is carried out. The conversion from series to parallel is covered at section 3, with the details of the GPU implementation strategy and results in section 4. Finally, the conclusions are summarized at section 5.

2. PARAMETRIC SOS-CHAIN MODEL

As commented, at [9] an IIR parametric model of HRTFs was proposed and evaluated by one of the authors based on prior work carried out for loudspeaker equalization and crossover design [16]. Here a brief resume will be done, refer to [9] for details. Figure 2 shows the parametric model. It is formed by an initial delay line that models the time difference between the responses of left and right ears. The delay could be integer, or fractional for a more precise modeling [17]. Later, a series of SOS formed by a low-shelving filter SHL_1, and peak filters PK_i, all of them defined by its parameters (frequency f_i, gain g_i, and Q q_i). This is a common approach, splitting the model in a pure delay line for the ITD, and a minimum-phase IIR or FIR filter.

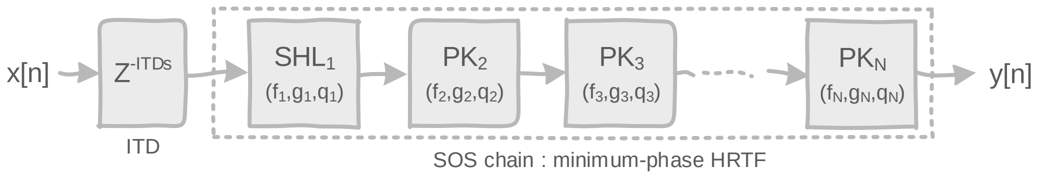

Fig. 2 – HRTFs parametric model as a SOS chain

The model is designed iteratively looking for the best set of parameters (f_i, g_i, q_i) of each SOS that minimizes the error areas between the HRTF to model and the response of the modeling filter. Figure 3a shows the error areas (A_1 to A_4), and the response with only the first two SOS. SHL_1 models the low frequency behavior of the HRTFs, and the PK_i the peaks and valleys [10], [11]. The SOS are designed one by one, trying to mimic the remaining biggest error area A_i. The parameters are found using a combination of a direct search method for getting the initial values of the parameters, and a second stage that performs a heuristic optimization of them. The cost function to minimize works in a double logarithmic domain, in frequency working with a discrete log-spaced frequency axis, and in magnitude using the absolute value of the magnitude in decibels. When the whole SOS are designed, a posterior post-optimization process is carried out, re-optimizing the parameter's values of several contiguous in frequency SOS that improves the interaction among them. Figure 3b shows the modeling filter with 12 SOS that matches the original HRTF using only the information of 36 parameters, instead of the 200 coefficient length of the original HRTF used [18].

Once modeled one HRTF position, i.e. at 0º, instead of modelling again the next position (5º) from the scratch, it is designed evolving the parameters of the already modeled position using the post-optimization process [9]. Normally, peaks and valleys of the responses change a little from one position to the next one. Figure 3c shows the complete set of HRTFs for a specific elevation with responses separated 10 dB for clarity.

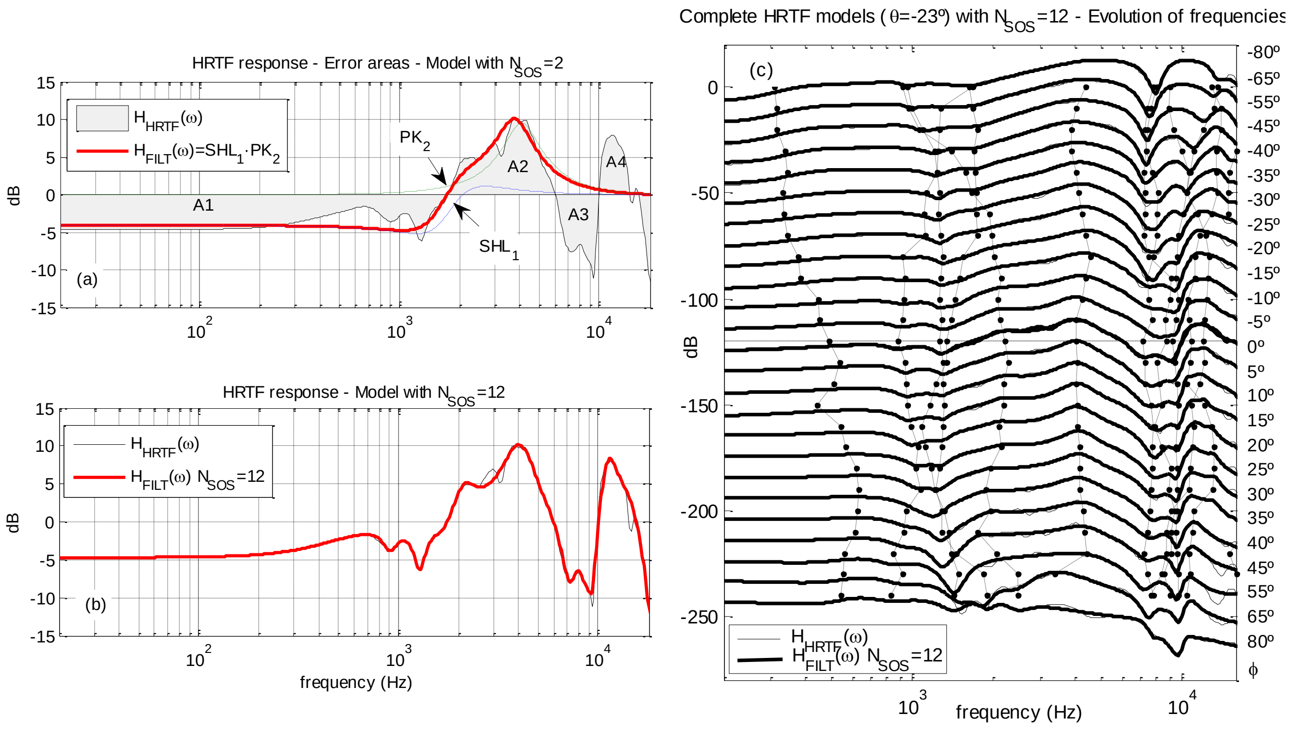

Fig. 3, from [9] – (a) Error areas, and modeling with 2 SOS.
(b) – Model obtained with a chain of 12 SOS.
(c) - Set of HRTFs and its models. Evolution of the frequencies

The vertical lines display the evolution of the frequencies of the SOS, which follow the frequencies of peaks and valleys of the responses. This approach let a simple interpolation procedure of HRTFs just doing an interpolation of the parameters, allowing to decrease the number of modeled positions, and achieving a reduction of one order of magnitude in the complete database size.

3. CONVERSION TO A PARALLEL SOS MODEL

The previous parametric HRTF model performed with the SOS chain of Figure 2, gives satisfactory results, but it is not well suited to be implemented in a GPU architecture, where a parallel implementation is preferred, since it allows splitting the computation among the multiple GPU cores [15].

The simplest (and equivalent filter implementation) is performing a series to parallel conversion by classical partial-fraction-expansion [19]. Once designed the model as a series of *N* SOS [9] defined by its coefficients b_{0i}, b_{1i}, b_{2i}, a_{1i} and a_{2i} , then a transformation to parallel is done finding the new coefficients *K* and b'_{0i}, b'_{1i}, a'_{1i} and a'_{2i} for the new parallel *N* SOS implementation of Figure 4. The ITD block with the delay line remains equal. By this way, now it is possible to do the calculus of each SOS-i in parallel, due to now, there is no data dependency between the different SOS, and finally do the global addition of the output of the SOS. This series-to-parallel conversion can be carried out offline, generating the new parallel HRTF data-base.

$$H(z) = \prod_{i=1}^{N} \frac{b_{0i} + b_{1i} \cdot z^{-1} + b_{2i} \cdot z^{-2}}{1 + a_{1i} \cdot z^{-1} + a_{2i} \cdot z^{-2}} = K + \sum_{i=1}^{N} \frac{b'_{0i} + b'_{1i} \cdot z^{-1}}{1 + a'_{1i} \cdot z^{-1} + a'_{2i} \cdot z^{-2}}$$

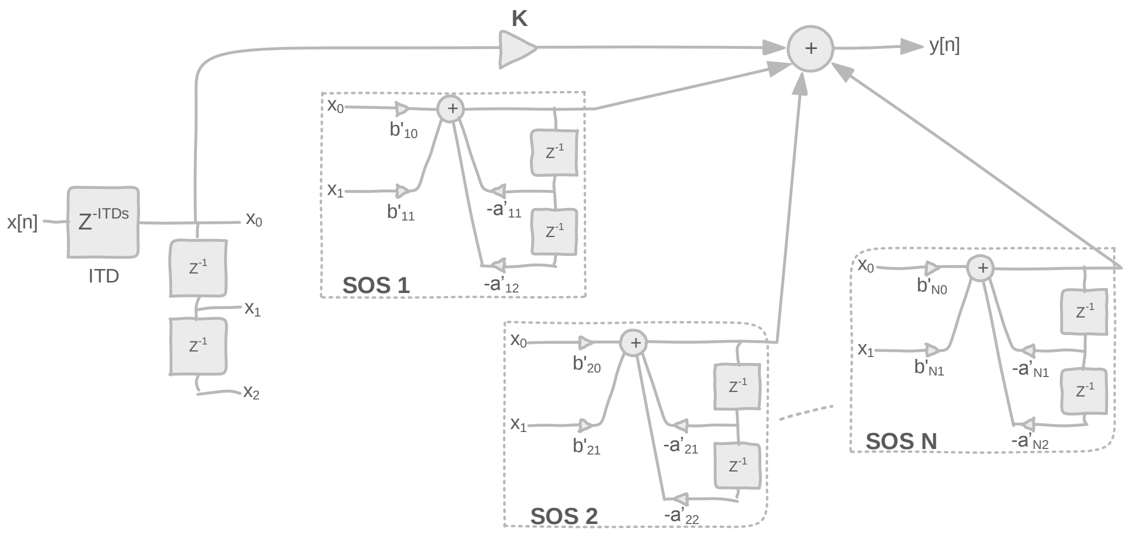

Fig. 4 – Parallel implementation by partial-fraction-expansion.

As an example, let's perform a model [9] with a chain of 12 SOS, shown at Figure 5a in blue with the thick line. Obtaining the partial-fraction-expansion, the individual responses of each SOS is displayed with the black thin lines. The complex addition of the responses, rebuild exactly the original chain model as expected. Now the phase between the SOS is relevant, as opposite with the series implementation.

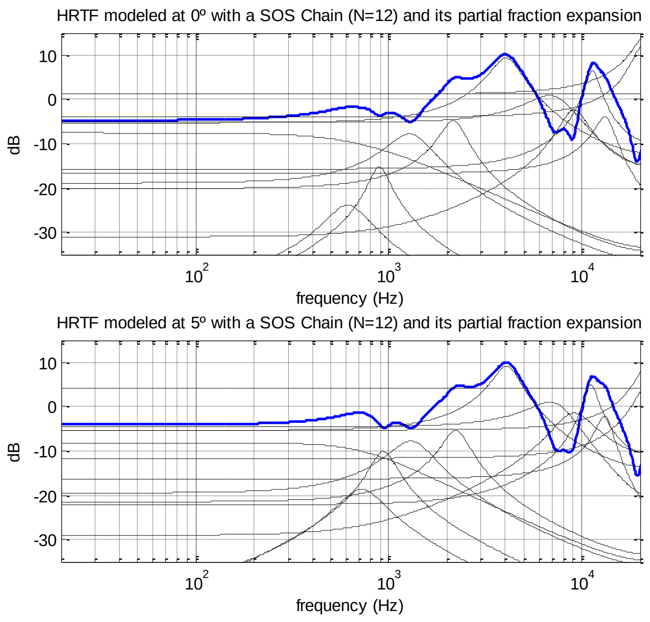

Fig. 5 – Partial-fraction-expansion with 12 SOS at two consecutive positions.

Figure 5b shows the HRTF model, again with 12 SOS, but at the next angle (before at 0º, now at 5º), and the individual responses of the partial-fraction-expansion. Some of the SOS responses are quite similar, but other start having differences of several decibels. This could difficult the interpolation procedure of [9]. In this case, it will be better to obtain the interpolated parametric SOS chain model, and then the new equivalent parallel one. Another approach is parameterizing the angles and radios of the poles, and the radios and gains of the zeros, and then interpolate them. Anyway, an equivalent parallel filter that matches the original one is obtained, that is better suited for its implementation in GPU architectures.

There are other options for creating a parallel filter to model the HRTFs, like the previous work on parallel filter banks by Bank with success [15], [20], [21] but without the parametric approach.

4. GPU IMPLEMENTATION

Compute Unified Device Architecture (CUDA) is a software programming model that allows the use of GPUs for applications beyond graphics rendering. GPUs have the potential of highly parallel data processing. The recent Nvidia GPU Kepler architecture [22] is composed of multiple Stream Multiprocessor (SMX), where each SMX consists of 192 pipelined cores per SMX. A GPU device has a large amount of off-chip device memory (*global-memory*) and a limited amount of fast on-chip memory (*shared-memory*).

The code to be executed on the GPU concurrently by multiple elementary processes, called threads, is written on a kernel function. The threads are grouped in Thread Blocks (TB). Before launching the GPU code, the programmer must define the number of TBs and its size (i.e., the number of threads). This is important, since only the threads that belong to the same TB can share data through *shared-memory*. More details can be found in [22].

If we want to render *N* sound sources in a binaural system, we need to compute 2*N* IIR filters concurrently, for the left and right channel. For performance and quality reasons, each HRTF filter will be designed with 16 SOS. Thus, we launch *N* TBs to run the CUDA kernel. Each TB is composed of 32 threads where each thread computes the parallel section SOS-i shown in Fig 6.

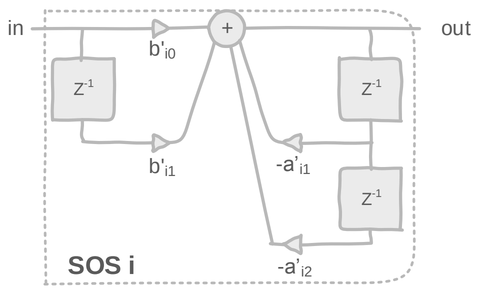

Fig. 6 – Section that is processed by one thread of the GPU.

The first 16 threads of the TB are devoted to compute the 16 sections that correspond to the left channel, whereas the second 16 threads of the TB compute the 16 sections that correspond to the right channel.

The TB that is used in this implementation is composed of two-dimensional threads (*threadIdx.x*, *threadIdx.y*), where *threadIdx.x* $\in$[0 15], and *threadIdx.y* $\in$\{0;1\}. Since there is a delay between samples that are processed for the left channel and the right channel, the variable x_0 in the GPU implementation is defined as:

$$x_0 = x_0 - \textit{threadIdx.y}*\text{ITD}$$

Thus, each TB computes two IIR filters (left and right). A thread inside the TB computes one section and stores its result in the *shared-memory*. Then, a synchronization barrier is set in order to wait that all the threads of the TB have finished. After that, the reduction algorithm described by Harris [23] is implemented. It consists in summing up in a parallel way all the values of a vector that is stored in the *shared-memory*. Finally, the coefficient $b'_{1,1}$ (K) that are shown in Fig. 4 is summed. Fig. 7 illustrates the described operations for one input sound source. It is important to highlight that the *N* TBs of the implementation are launched concurrently and executed in parallel.

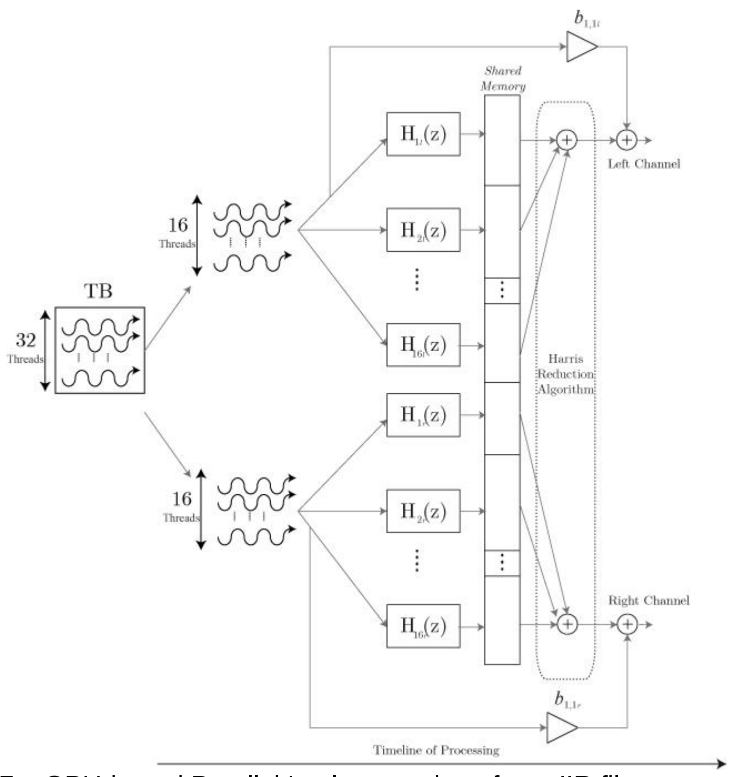

Fig. 7 – GPU-based Parallel Implementation of one IIR filter processing.

We tested our GPU-based implementation on an Nvidia Tesla K20Xm that is based on the Kepler architecture and is composed of 14 SMXs. We used a standard audio card at the laboratory. The audio card uses the ASIO (Audio Stream Input/Output) driver to communicate with the CPU and provides 8, 16, 32 and 64 samples per channel every 0.18 ms, 0.36 ms, 0.72 ms and 1.45 ms, respectively (sample frequency fs=44100 Hz). Figure 8 shows the maximum number of sound sources that can be computed under real-time conditions.

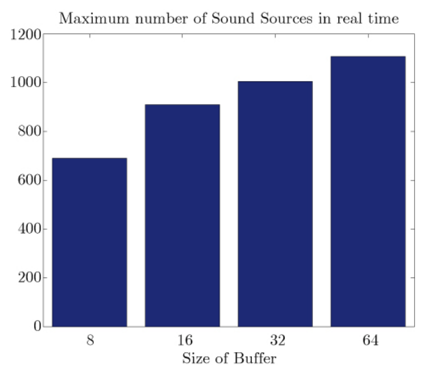

Fig. 8 – Maximum number of sound sources computed under real-time conditions

The proposed GPU-based implementation can run 690 sound sources in real time with 0.18 ms latency. In case latency is 1.45 ms, then 1108 sources can be rendered in real time. With latencies of 0.72 and 0.36, the system is able to manage 910 and 1005 sound sources respectively.

5. CONCLUSIONS

An efficient implementation in GPUs of parametric HRTFs models has been presented and validated. Starting from previous work of the authors in HRTFs modeling with SOS chains, an equivalent parallel model has been developed in order to be adapted to the parallel nature of GPUs. The implemented solution allows to decrease the computational cost for using HRTF techniques moving the calculus from the traditional CPU or DSP to the GPU, permitting high accuracy HRTF models with a great number of moving sound sources in real time.

6. ACKNOWLEDGEMENTS

The Generalitat Valenciana has supported this work under the project GV/2013/134. Part of this work has been also founded by the Spanish Ministry of Economy and Competitiveness supported under the projects TEC2012-37945-1091C02-01/02.

7. REFERENCES

[1] F. L. Wightman and D. J. Kistler, “Headphone simulation of free-field listening. II: Psychophysical validation”, J. Acoustical. Society of America, vol. 85, pp. 868–878, 1989.
[2] H. Møller, M. Sørensen, D. Hammershøi, and C. Jensen, “Head-related transfer functions of human subjects,” J. Audio Engineering Society, vol. 43, no. 5, pp. 300–321, May, 1995.
[3] J. Huopaniemi, “Virtual acoustics and 3-D sound in multimedia signal processing,” Ph.D. dissertation, Helsinki Univ. of Technology, 1999.
[4] V. R. Algazi and R. O. Duda, “Headphone-based spatial sound”, IEEE Signal Processing Magazine 28, 33–42 2011.
[5] J. Sandvad and D. Hammershøi, “Binaural auralization. comparison of FIR and IIR filter representation of HIR’s,” in Proc. 96th Audio Eng. Soc. Convention, Amsterdam, preprint 3862, Mar. 1994.
[6] Y. Haneda, S. Makino, Y. Kaneda, and N Kitawaki, “Common-acoustical-pole and zero modeling of Head-Related Transfer Functions,” IEEE Transactions on Speech and Audio Processing, vol. 5, no. 3, pp. 188-196, May 1997.
[7] J. Mackenzie, J. Huopaniemi, V. Välimäki, and I. Kale, ”Low-order modeling of Head-Related Transfer Functions using Balanced Model Truncation,” IEEE Signal Processing Letters, vol. 4, no. 2, pp. 278-280, Feb. 1997.
[8] E. A. Durant and G. H. Wakefield, “Efficient model fitting using a Genetic Algorithm: pole-zero approximations of HRTFs,” IEEE Transactions on Speech and Audio Processing, vol. 10, no. 1, pp. 18-27, Jan. 2002.
[9] G. Ramos and M. Cobos, “Parametric head-related transfer function modeling and interpolation for cost-efficient binaural sound applications”, J. of the Acoustical Society of America, vol 134, no. 3, pp. 1735-1738, Sep. 2013 doi:10.1121/1.4817881.
[10] R. Bristow-Johnson, “Cookbook formulae for audio EQ biquad filter coefficients”, on-line publication: http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt
[11] U. Zölzer, Digital Audio Signal Processing, 2nd Edition, Ed. John Wiley & Sons, 2011.
[12] F. Freeland, L. W. P. Biscainho, and P. S. R. Diniz, “Interpositional transfer functions for 3D-sound generation,” J. Audio Eng. Soc., vol. 52, no. 9, pp. 915–930, Sep. 2004.
[13] B. Carty, V. Lazzarini, “Frequency-domain interpolation of empirical HRTF data”, in Proc. 126th Audio Engineering Society Convention, preprint 7725, , May 2009.
[14] J. A. Belloch, M. Ferrer, A. Gonzalez, F. Martinez-Zaldivar, and A. M. Vidal, “Headphone-based virtual spatialization of sound with a GPU accelerator,” J. Audio Eng. Soc, vol. 61, no. 7/8, pp. 546–561, 2013.
[15] J. A. Belloch, B. Bank, L. Savioja, A. Gonzalez, and V. Välimäki, “Multi-channel IIR Filtering of Audio Signals Using a GPU,” in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, May 2014..
[16] G. Ramos and J. J. Lopez, “Filter design method for loudspeaker equalization based on IIR parametric filters,” J. Audio Eng. Soc. 54, 1162–1178, Dec. 2006.
[17] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, “Splitting the unit delay-Tools for fractional delay filter design”, IEEE Signal Process. Mag. vol. 13, pp. 30–60, 1996.
[18] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, “The CIPIC HRTF database”, in Proceedings of the 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp, 99–102, 2001.
[19] A. V. Oppenheim, R. W. Schafer, Discrete-Time Signal Processing. Ed. Prentice Hall, 1989.
[20] B. Bank, "Audio Equalization with Fixed-Pole Parallel Filters: An Efficient Alternative to Complex Smoothing," The Journal of the Audio Engineering Society, Vol. 61, Iss. 1/2, pp. 39-49 (2013)
[21] B. Bank, G. Ramos, “Improved Pole Positioning for Parallel Filters Based on Spectral Smoothing and Multi-Band Warping”, IEEE Signal Processing Letters, vol. 18, no. 5, pp. 299-302 (2011)
[22] Nvidia CUDA developer zone: https://developer.nvidia.com/cuda-downloads
[23] Nvidia CUDA Optimizing Parallel Reduction in CUDA NVIDIA: http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf