GPU Implementation of a Real-time WFS System with Room Compensation Using Fractional Delay Filters

Jose A. Belloch*, *Member, IEEE,* Alberto Gonzalez, *Senior Member, IEEE,* Antonio M. Vidal, Miguel Ferrer, *Member, IEEE,* and Vesa Välimäki, *Fellow Member, IEEE*

Abstract

Wave Field Synthesis (WFS) is a spatial audio reproduction system that provides an accurate spatial sound field in a wide area. This sound field is rendered through a large number of loudspeakers to emulate virtual sound sources. The emergence of the GPUs as a massive signal co-processor allows to increase the capabilities of these systems. Rendering movements of the virtual sound sources in real time is an important issue when a defined trajectory with a specific resolution must be performed. We tackle this issue by adding an additional filtering stage composed by multiple Fractional Delay Filters that allow to render with more accuracy a sound source in the system. On the other hand, the interaction of the loudspeaker array with the listening room can generate new echoes that avoid a correct generation of the synthesized sound field. Compensating echos is carried out with an inverse filter bank which further increases the computational requirements of the WFS system. This paper analyzes the performance of a real WFS system that is implemented on a GPU according to: length of room compensation filters, and trajectory resolutions of the sound source movements. Our WFS system is able to synthesize movements of 21 virtual sound sources in real time with a trajectory resolution of 0.007 m, reaching a maximum speed of 0.6 m/s, and using 9216 room compensation filters of more than 3000 coefficients each one.

J. A. Belloch is with the Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I, 12071, Castellón, Spain.

* Corresponding author. E-mail: jbelloch@uji.es.

A. Gonzalez and M. Ferrer are with the Institute of Telecomunications and Multimedia Applications, Universitat Politècnica de València, 46022, Valencia, Spain.

A. Vidal is with the DSIC department, Universitat Politècnica de València, 46022, Valencia, Spain.

V. Välimäki is with the Dept. of Signal Processing and Acoustics, Aalto University, Espoo, Finland.

Index Terms

Audio systems, signal synthesis, parallel architectures, parallel processing

I. INTRODUCTION

In the last decades, there has been increasing interest in listening experience and more specifically in spatial audio rendering. One of the spatial audio systems available today is the Wave Field Synthesis (WFS), where sound field is synthesized in a wide area by means of arrays of loudspeakers, which are referred to as secondary sources. WFS is usually implementing discrete-time signal processing and is able to reproduce complex auditory scenes consisting of multiple acoustic objects, which are generally denoted as primary or virtual sources. The WFS concept was introduced at the Delft University of Technology in 1980's. Berkhout carried out first researchers in this field [1], [2], which were followed by different dissertations such as [3], [4], [5], [6], [7].

One of the problems to put WFS in practise is related to the interaction of the array with the listening room. The listening room introduces new echoes that are not included in the signal to be reproduced, thus altering the synthesized sound field and reducing the spatial effect. One block that can be added to this system is a Room Compensation (RC) block. The purpose of this block is to minimize the undesirable interaction of the array with the listening room. A common RC block is based on a multichannel inverse filter bank that corrects the room effects at selected points within the listening area, such as those in [8] and [9]. This formula is validated by [10], where it is presented meaningful improvements in the acoustic field when a RC block is applied to a WFS system. However, the application of this spatial audio system (WFS + RC) in real environments (theaters, cinemas, etc.) requires a real-time solution which demands high computational requirements.

One special situation occurs when it is required to move a sound source through a specific trajectory. In a practical WFS system that is implemented on a discrete time can imply to delay a signal a number of samples that is not an integer value. To this end, we propose the use of the fractional delay filters, [11] [12] for the rendering of the audio signals in a WFS system. These filters are used to interpolate a signal sample that is between two sampled values. Applications of fractional delay filtering in audio signal processing include digital audio effects [13], [14], physical sound synthesis [15], [16] or sound reproduction systems [17].

A large-scale WFS system that uses massive filtering demands high computational needs. An approach to solve the computational needs consists in performing all the audio processing tasks in the Graphics

Processing Unit (GPU), i.e by using the GPU as a co-processor. In fact, GPU computing has already been applied to different problems in acoustics and audio processing. Applications include room acoustics modelling [18], [19], [20], [21], computer-music synthesis using additive synthesis [22], [23], full 3-D model of drums in a large room [24], sliding phase vocoder [25], beamforming [26], audio rendering [27], [28], [29], multichannel IIR filtering of audio signals [30], dynamic range reduction using multiple allpass filters [31], and adaptive filtering [32], [33], [34], among others.

Up to now, there have been several studies aiming at implementing a WFS system. In [35], it is presented a WFS implementation that benefited of a time invariant preprocessing in order to reduce CPU load. In [36], Theodoropoulos et al. propose a minimal processor architecture adapted to WFS-based audio applications. They estimated that their system could render in real time up to 32 acoustic sources when driving 64 loudspeakers. The same authors presented in [37] a WFS implementation on different multi-core platforms, including a GPU-based implementation that achieved more than 64 sources when driving 96 loudspeakers. They concluded that GPUs are suitable to build inmersive-audio real-time systems.

In [38], the authors present also a GPU-based implementation of WFS. In comparison to them, the present work increase the audio processing by dealing with Room Compensation and Fractional Delays. Moreover, we design our application to achieve maximum performance on GPUs making use of the Kepler architecture GK110 [39]. This architecture can be found on the Tegra K1 (TK1) systems-on-chip (SoC), embedded in the Jetson development kit (DevKit) [40], and it is becoming widespread in current mobile devices such as Google's Nexus 9 tablet [41]. Thus, the proposed implementation can be successfully adapted to work properly on GPUs that are currently embedded in mobile devices.

Our previous work [42] presented a preliminary approach that performed a WFS system using a fixed length of sample buffers and a fixed length of room compensation filters. Comparing to previous work [42], this current approach uses uniformly-partitioned FFT-based convolution in the frequency-domain in order to convolve room compensation filters that have large number of coefficients, with short audio sample buffers.This contribution is important, since it allows to reduce the latency of the system independently of the length of the room compensation filters. Moreover, we use fractional delay filters, which allows to synthesize a more accurate virtual sound source position and define sound source trajectories that demand specific resolutions. Finally we must highlight that all the audio processing that requires this new implementation is performed by one of the latest Kepler-based family of GPUs (Tesla K20Xm).

The paper is structured as follows. Next section briefly describes some characteristics of the GPUs, and highlights the features to take into account when a real-time WFS prototype is implemented on GPUs. An overview of the theory of the WFS is described in Section III. Section IV presents the main features

of our WFS implementation. Section VI enumerates different kinds of fractional delay filters and assesses which fractional delay filter fits better with a WFS system. Detailed GPU-based implementation of the WFS system is described in Section VII. Performance of the WFS system is devoted to Section VIII. Finally, section IX is dedicated to some concluding remarks.

II. A GPU IN A REAL-TIME WFS SYSTEM

Dealing with real-time audio applications on GPU requires a basic understanding of the GPU programming features. This section provides a basic description of the GPU data flow and some relevant issues that must be taken into account when programming a real-time WFS application.

A. GPU and CUDA

Following Flynn's taxonomy [43], from a conceptual point of view, a GPU can be considered to be a Single Instruction Multiple Data machine (SIMD), i.e, a computer in which a single set of instructions is executed on different data sets. Implementations of this model usually work synchronously, with a common clock signal. An instruction unit sends the same instruction to all of the processing elements, which execute this instruction on their own data simultaneously. The last generations of GPUs are composed by multiple Stream Multiprocessor (SM), where each SM consists of eight pipelined cores (SP) if compute capability is 1.2 or 1.3 (Tesla architecture), or 32 pipelined cores if it is 2.0 (Fermi architecture), or even 192 pipelined cores if it is 3.0 or 3.5 (Kepler architecture) [44].

A GPU device has a large amount of off-chip device memory (*global-memory*) and a fast on-chip memory (*shared-memory*, *registers*). As its name indicates, the *shared-memory* is normally used when multiple threads must share data. There are also read-only cached memories called *constant-memory* and *texture-memory*. The first memory is optimized for broadcast i.e. when all the threads read the same memory location while the second one is more oriented to graphics.

An important aspect when accessing *global-memory* is to do it in a *coalesced* way, which can reduce meaningfully the memory-access time. Coalescing means that the threads are writing/reading into a small range of memory addresses having a certain pattern. For example, considering an `array` pointer to *global-memory* and `idx` as the identification of a thread, if thread `idx` writes/reads to address `array[idx]` and thread `idx+1` to address `array[idx+1]`, we achieve good coalescing.

GPU devices of compute capability 2.x and greater come with an L1/L2 cache hierarchy that is used to cache *global-memory*. Cache of level L1 is located on-chip memory. The same occurs to the *read-only cache* that is only present in GPU devices of compute capability 3.x. This *read-only cache* plays an

important feature, since it supports full speed unaligned memory access patterns among other scenarios, what reduces the penalization of the uncoalesced access to *global-memory*. Fig. 1 shows how the GPU architecture is organized.

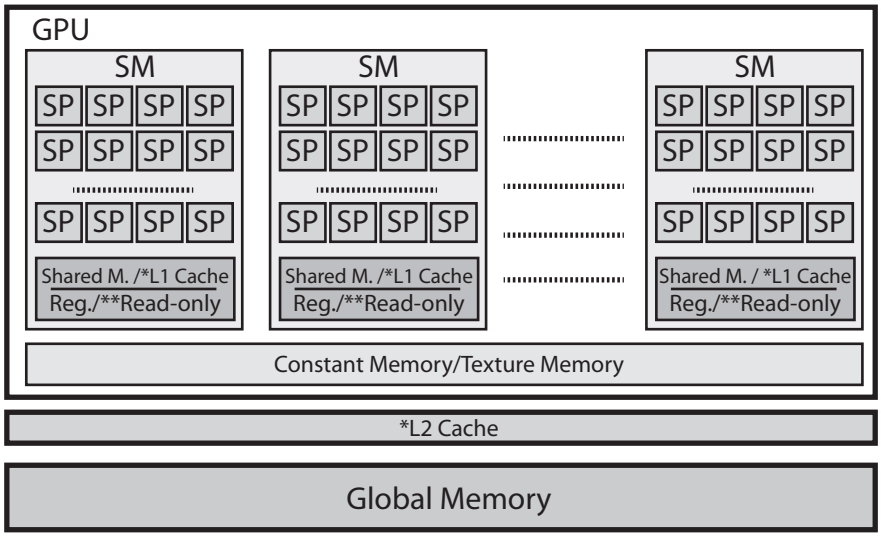

Fig. 1. A GPU has multiple Stream Multiprocessor (SM) that are composed of multiple pipelined cores (SP). The number of SPs depends on the compute capability and the number of SMs depends on the kind of the device. A GPU device has off-chip device memories and on-chip memories *(in devices with compute capability 2.x and 3.x) **(only in devices with compute capability 3.x).

CUDA technology is an environment based on C language which allows the development of software intended to solve high complexity computational problems efficiently [44]. This software takes profit from the high amount of execution threads which are available in GPU. In CUDA, the programmer defines the kernel function where the code to be executed on the GPU is written. A grid configuration, which defines the number of threads and how they are distributed and grouped, must be built into the main code.

B. Real-Time processing of a WFS system on GPU

The WFS system used for implementing the WFS rendering using the GPU as a coprocessor is located at the Universitat Politècnica de València (UPV) and belongs to the Audio and Communications Signal Processing Group [45]. This system is composed by 96 loudspeakers (N=96) that are positioned using an octogonal geometry. The separation between two loudspeakers is 18 cm. Fig.2 shows the WFS system at the laboratory.

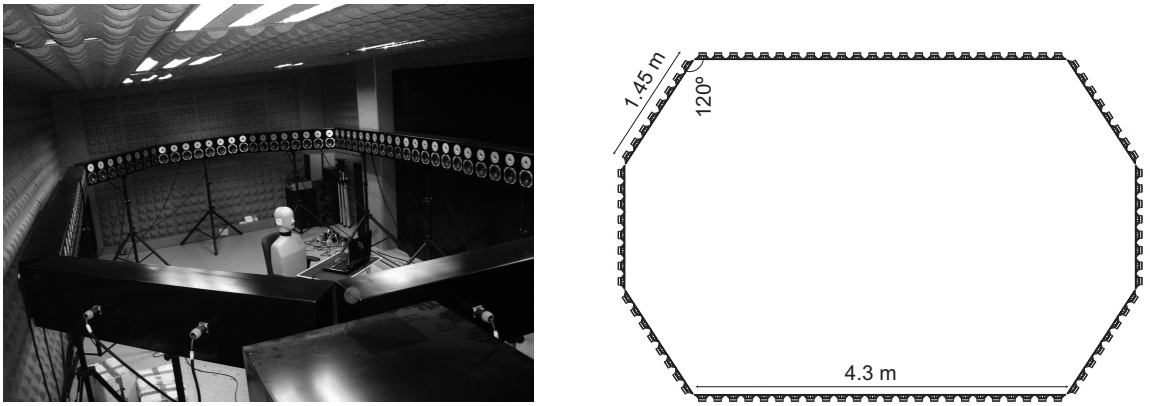

Fig. 2. Configuration of the array at the laboratory of the GTAC at UPV

The N=96 loudspeakers are connected to four MOTU 24I/O audio cards that uses the ASIO (Audio Stream Input/Output) driver to communicate with the CPU. The ASIO driver works with blocks of samples of size L and at sampling rate f_s. Thus, each Lf_s seconds, the audio card requires N output-data buffers of size L to be reproduced thorough the loudspeakers. In fact, this time is known in the literature [29] as t_{buff} and is independent of the number of loudspeakers and virtual sound sources of the WFS system. In contrast, the processing time t_{proc} depends on M and N, being M and N the number of virtual sound sources and of loudspeakers in the WFS system, respectively. Thus, t_{proc} includes the time spent on data transfers between GPU and CPU and the time employed for the computation on the GPU (cuda kernels). These data transfers are carried out by using PCI-e X 16 bus, which has a rate $\approx$ 8 GB/s. The WFS system always works in real time as long as $t_{\mathrm{proc}} < t_{\mathrm{buff}}$. When this condition is no longer valid, this does not mean that the application stops working, since it could work as an off-line application, i.e. recording the audio samples in order to reproduce them afterwards.

III. WFS FUNDAMENTALS

Wave Field Synthesis is a sound reproduction method, based on fundamental acoustic principles [2], [1]. It enables the generation of sound fields with natural temporal and spatial properties within a volume or area bounded by secondary sources (arrays of loudspeakers, see Fig.2). This method offers a large listening area with uniform and high reproduction quality.

The theoretical basis of WFS is given by the Huygens' principle. According to this, the propagation of a wave front can be described by adding the contribution of a number of secondary point sources distributed along the wave front. This principle can be used to synthesize acoustic wave fronts of an

arbitrary shape.

A synthesis operator for each loudspeaker can be derived. The general 3D solution can be transformed into the 2-D solution, which is sufficient for reconstructing the original sound field in the plane of listening [46], [3], [4]. For that purpose a linear array of loudspeakers is employed to generate the sound field of virtual sources.

Following a model-based rendering in which point sources and plane waves are used [47], the field rendered by a sound source m at point R within the area surrounded by N loudspeakers can be expressed as

$$P(\mathbf{x}_R, \omega) = \sum_{n=0}^{N-1} Q_n(\mathbf{x}_m, \omega) \frac{e^{\frac{-j\omega r_{nR}}{c}}}{r_{nR}}, \tag{1}$$

where c is the speed of the sound, $\mathbf{x}_m$ is the position of the virtual sound m, $\mathbf{x}_R$ is the position of the point R, and r_{nR} is the distance between the nth loudspeaker and the point R ($r_{nR} =\mid \mathbf{x}_n - \mathbf{x}_R \mid$).

The driving signal of the nth loudspeaker is represented by $Q_n(\mathbf{x}_m, \omega)$, which is computed as

$$Q_n(\mathbf{x}_m, \omega) = S(\omega)\sqrt{\frac{j\omega}{2\pi c}} K \frac{1}{\sqrt{r_{mn}}} \cos(\theta_{mn}) e^{\frac{-j\omega r_{mn}}{c}}. \tag{2}$$

where K is a geometry dependent constant, $r_{mn} =\mid \mathbf{x}_m - \mathbf{x}_n \mid$, and $\mathbf{x}_n$ is the position of the loudspeaker n. Figure 3 shows the geometry of the system, where θ_{mn} is the angle between the line that connects $\mathbf{x}_m$ and $\mathbf{x}_n$, and the normal vector $\mathbf{n}$ of the loudspeaker n. The piano represents the sound source m. The driving signal (2) consists of several elements that have different functionalities. The term $S(\omega)$ is the frequency-domain characteristics of the source signal, while the term

$$H(\omega) = \sqrt{\frac{j\omega}{2\pi c}}, \tag{3}$$

represents a filtering operation that is independent of the position of the virtual source. In [48], it is referred as a WFS pre-equalization filter that represents a lowpass filter with a constant slope of 3 dB/octave if loudspeaker is considered a monopole secondary source, while it forms a highpass with a magnitude increase of 3 dB/octave in case of dipole secondary sources. An important contribution in (2) is

$$a_{mn} = \frac{K}{\sqrt{r_{mn}}} \cos(\theta_{mn}) \tag{4}$$

that denotes an amplitude factor that depends on the positions of the sound source m, and the loudspeaker n. Finally, the $e^{\frac{-j\omega r_{mn}}{c}}$ represents the phase shift corresponding to a time delay that depends on the distance between the virtual sound source m and the loudspeaker n.

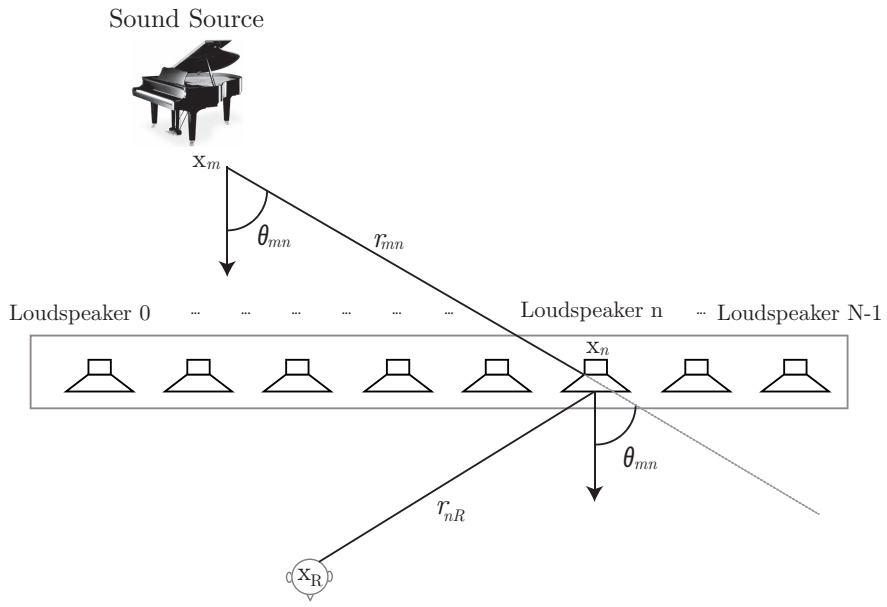

Fig. 3. Geometry of a WFS system where it is appreciated: the sound source m (piano), the N loudspeakers, and the distances among sound source, loudspeakers and a listener.

The driving signal shown in (2) is expressed also in time domain as

$$q_n^m(t) = a_{mn} \cdot s_m(t) * h(t) * \delta(t - \frac{\mid \mathbf{x}_m - \mathbf{x}_n \mid}{c}). \tag{5}$$

where $*$ denotes the convolution operator, $s_m(t)$ is the signal of sound source m, and $h(t)$ is the inverse fourier transform of $H(\omega)$ in (3).

In a multi source system composed of M virtual sound sources, the loudspeaker driving signal of the nth loudspeaker is

$$q_n(t) = \sum_{m=0}^{M-1} q_n^m(t). \tag{6}$$

In a discrete-time signal processing system with sampling frequency f_s, expressions (5) and (6) turn to

$$q_n[k] = \sum_{m=0}^{M-1} a_{mn} \cdot s_m[k] * h[k] * \delta[k - \tau_{mn}]. \tag{7}$$

where k is the sample index, and τ_{mn} is the delay in number of samples that must be computed in the discrete signal $q_n^m[k]$.

$$\tau_{mn} = \frac{\mid \mathbf{x}_m - \mathbf{x}_n \mid}{c} f_s \tag{8}$$

A. Room Compensation in a WFS system

As introduced in Section 1, the interaction of the driving signals with the listening room can deteriorate the localization properties that a WFS system has. The synthesized sound field can be altered by new echoes that are introduced by the listening room and that reduce the spatial effect. Lopez et al. designed and validated in [10], [49] a multichannel inverse filter bank that corrects these room effects at selected points within the listening area. In a WFS system composed of N loudspeakers, this implies to add N^2 FIR filters to the system, increasing its computational demand. Equation 9 shows the operations that are carried out in a multichannel inverse filter bank with every driving signal. The final signal to be reproduced by the nth loudspeaker $y_n(t)$ is a combination of all the filtered signals, as illustrated in Fig.(4), where the filter $f_{0n}(t)$ goes from the driving signal $q_0(t)$ to the loudspeaker n.

$$y_n(t) = \sum_{j=0}^{N-1} q_j(t) * f_{jn}(t). \tag{9}$$

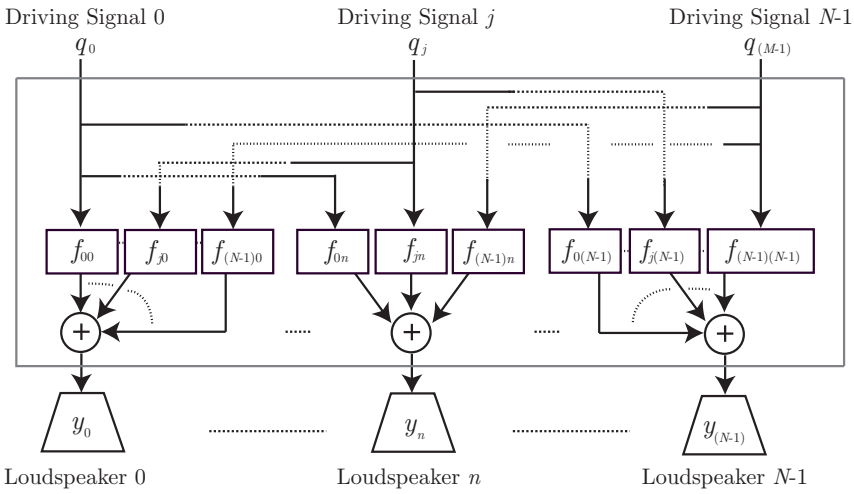

Fig. 4. Multichannel inverse filter bank, where every driving signal is convolved by N filters. The signal that is reproduced by a loudspeaker is a combination of all the filtered signals.

The calculation of the inverse filters can be carried out in a setup stage, since the main room reflections can be considered invariant for each specific room. Different methods have been proposed to obtain the bank of correction filters. There are methods that compute an approximate solution in frequency domain using FFT [50]. However, we use in this work a method that guarantees a minimal square error solution in time domain [51] for computing the correction filters f_{jn}. A detailed description of the operation to carry out for the computation of the filters is achieved in [10].

IV. Main Features for implementing a WFS system

This section summarizes the main features that our WFS implementation uses in order to synthesize properly the driving signals regarding two important aspects: audio signal processing and sound source movements.

A. Audio Signal Processing

As stated in Section 2.1, our WFS system is composed by several multichannel audio cards that provide audio buffers each Lf_s seconds, where L is the size of the buffer in samples. We denote $\mathbf{s}_{\mathrm{buff}_m}$ to the input-data buffer of the L samples of the sound source m, and $\mathbf{y}_{\mathrm{buff}_n}$ to the output-data buffer of the L samples that feeds the loudspeaker n.

Previous work [42] implemented a WFS system based on overlap-save technique in the frequency domain. To this end, processing audio buffers of size $2L$ with a 50% overlap were used. Thus, the processing of the audio buffers was composed by the current input-data buffer and the previous one that came from the audio card. In that implementation, only $2L$ samples per source were involved in the processing. The equation that was carried out for computing the output buffers was:

$$\mathbf{Q}^m_{\mathrm{buff}_n} = (\mathbf{S}_{\mathrm{buff}_m} \circ \mathbf{H} \circ \mathbf{e}^{\frac{-\mathbf{j2\pi[0\ldots(2L-1)]\tau_{mn}}}{\mathbf{2L}}}) \cdot a_{mn}. \tag{10}$$

where $\mathbf{Q}^m_{\mathrm{buff}_n}$ and $\mathbf{S}_{\mathrm{buff}_m}$ were $2L$ frequency bins that corresponded to buffers of the driving signal of loudspeaker n and the source signal of source m. Parameter $\mathbf{S}_{\mathrm{buff}_m}$ was obtained after the $2L$-FFT transformation of $\mathbf{s}_{\mathrm{buff}_m}$. Symbol $\circ$ denoted an element-wise multiplication between the complex vectors. Parameter $\mathbf{H}$ was a $2L$-FFT transformation of the filter h that is shown in (7).

Finally, the driving signals for each loudspeaker were computed, and afterwards, the multichannel inverse filter bank of Fig.(4) was applied to these signals. Previously to the computation of the driving signals, the $2L$-FFT transformations were applied to all the filters f_{jn} that compose the multichannel inverse filter bank. Thus, all the processing was carried in the frequency domain. Equations (11) and (12) summarize these operations.

$$\mathbf{Q}_{\mathrm{buff}_n} = \sum_{m=0}^{M-1} \mathbf{Q}^m_{\mathrm{buff}_n}. \tag{11}$$

$$\mathbf{Y}_{\mathrm{buff}_n} = \sum_{j=0}^{N-1} \mathbf{Q}_{\mathrm{buff}_j} \circ \mathbf{F}_{jn}. \tag{12}$$

Note that the coefficients values of the filter are designed mainly attending to setup of the loudspeakers in the system [48].

For this presented work, we reduce the number of $2L$-FFT transformations by convolving filter h, that is independent of the position of the virtual sound source, with all the filters that compose the multichannel inverse filter bank following the equation

$$\widetilde{f}_{jn} = f_{jn} * h. \tag{13}$$

where $n, j \in [0, N-1]$. Thus, the only operations that must be carried out for our WFS implementation are: delay and weight the source signal.

$$q_n = \sum_{m=0}^{M-1} a_{mn} \cdot s_m * \delta[k - \tau_{mn}]. \tag{14}$$

The consequence of executing equation (13) previous to the real-time processing (off-line processing) implies that it is not necessary to carry out the delay in the frequency domain. Thus, for this WFS implementation, we perform the delay τ_{mn} and the weight a_{mn} of the sound signal in time domain.

B. Sound sources movements

The virtualization of the sound source movements is carried out by varying smoothly the virtual positions of sound sources over time. Fig. 5 shows a possible trajectory of the sound source in the WFS system (double dark arrow), where d_{AB} is the distance between the point A and point B. In each point of the trajectory, the variables a_{mn} and τ_{mn} have been computed. Table I shows the maximum change that is produced in the amplitudes and delays of the driving signals when a virtual sound source m is shifted in different steps of d_{AB}. This means that if this virtual sound source is being rendered in point A, and afterwards in point B, its a^A_{mn} and τ^A_{mn} have been radically change to a^B_{mn} and τ^B_{mn}. Second and third column of Table I show $\max \mid a^A_{mn} - a^B_{mn} \mid$ and $\max \mid \tau^A_{mn} - \tau^B_{mn} \mid$ for all the sound sources and loudspeakers in the WFS system.

The change in amplitudes remains practically stable as the virtual sound source shifts, however the delays vary substantially depending on the trajectory resolution that is represented in Table I by d_{AB}. For synthesizing the trajectory of the sound source with a resolution of fewer than 8 mm, it is required to carry out an interpolation technique that allows to produce delay values of fewer than one sample. In order to delay a signal a number of samples that is not an integer value, we propose the use of the fractional delay filters in a WFS system.

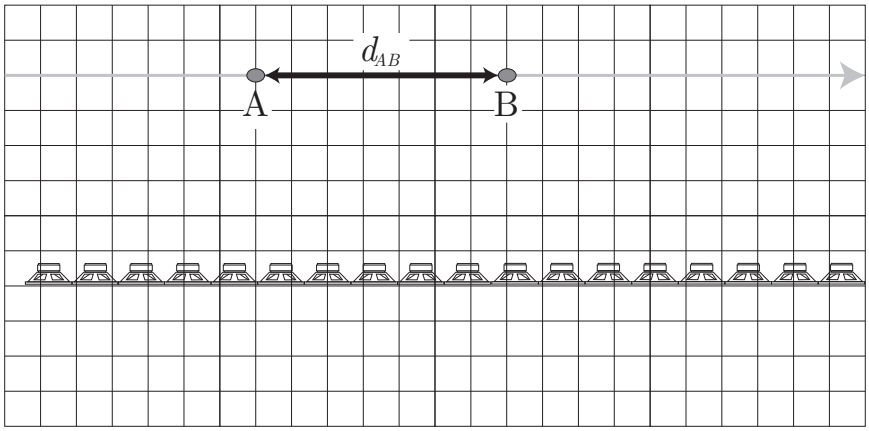

Fig. 5. Trajectory that follows a virtual sound source in a WFS system.

TABLE I

MAXIMUM CHANGES IN THE AMPLITUDES AND DELAYS IN THE DRIVING SIGNALS WHEN A VIRTUAL SOUND SOURCE IS SHIFTED IN STEPS OF d_{AB} (M).

d_{AB}	$\max \mid a_{mn}^{A} - a_{mn}^{B} \mid$	$\max \mid \tau_{mn}^{A} - \tau_{mn}^{B} \mid$
0.0001	0.0001	0.0126
0.0010	0.0014	0.1260
0.0025	0.0035	0.3149
0.0035	0.0048	0.4408
0.0050	0.0070	0.6298
0.0070	0.0090	0.8815
0.0075	0.0105	0.9447
0.0080	0.0112	1.0077

V. FRACTIONAL DELAYS

Computing τ_{mn} can imply to delay a signal a number of samples that is not an integer value. A common solution to this problem consists in rounding τ_{mn} to the nearest integer value. However, this can lead to acoustic artifacts [52].

The fractional delay filters allow to delay a digital signal by a fractional number of samples [11]. In [15], it is presented different techniques of fractional delay filters, such as linear interpolation, cubic interpolation, or Lagrange interpolation.

Linear interpolation is achieved by filtering the signal through a first order FIR filter whose equation

is

$$y[k-\alpha] = (\alpha - 1)y[k] + \alpha y[k-1]. \tag{15}$$

where α is a decimal number.

Cubic interpolation is similar to the linear interpolation and is achieved by filtering the signal through a third order FIR filter whose equation is

$$y[k-\alpha] = \sum_{j=0}^{3} h_{fd}[j]y[k-j]. \tag{16}$$

where

$$\begin{aligned} h_{fd}[0] &= -(1/6)(D-1)(D-2)(D-3), \\ h_{fd}[1] &= (1/2)D(D-2)(D-3), \\ h_{fd}[2] &= -(1/2)D(D-1)*(D-3), \\ h_{fd}[3] &= (1/6)D(D-1)(D-2), \end{aligned} \tag{17}$$

and $1 < D < 2$.

A more sophisticated technique is introduced in [53]. It consists of a modified design for fractional delay FIR filters, which is based on truncating a Lagrange Fractional Delay filter. This means to delete a number of coefficients at the beginning and the end of the coefficient vector of a prototype Lagrange Fractional Delay filter. This technique can be interpreted as a hybrid method that combines properties of the Lagrange and the truncated sinc Fractional Delay filters [15]. The coefficients design is computationally efficient and is based on a polynomial formula. The P-th order truncated Lagrange Fractional delay filter $h_{fd}[k]$ is obtained by casting off K_1 coefficients from each end of the T-th order prototype Lagrange Fractional Delay filter $h_L[k]$

$$h_{fd}[k] = \begin{cases} 0 & 0 \leq k \leq K_1 - 1 \\ h_L[k] & K_1 \leq k \leq P + K_1 \\ 0 & P + K_1 + 1 \leq k \leq T \end{cases}$$

$$h_L[k] = \prod_{p=0, p\neq k}^{T} \frac{D-p}{k-p}. \tag{18}$$

where $T > P$, K_1 is a positive integer ($K_1 > T/2$), and D is here a real number that depends on the fractional delay to achieve [11], [54].

In order to assess the effect of the fractional delays in a WFS system, we provide an objective comparison among three different WFS driving signals. The virtual sound source is composed in all

cases of an specific tone f and has a duration of 3 seconds (signals composed of $3f_s$ samples). These are the three different WFS driving signals generated from the virtual sound source:

$Q_{I,n}$ This signal is the reference. From the a_{mn} and τ_{mn} (m is equal to 1), it is computed

$$Q_{I,n} = a_{mn}\sin(2\pi\frac{f}{f_s}(k-\tau_{mn})); \tag{19}$$

where k is the sample index $k \in \{0,1,2,\ldots,3f_s-1\}$, $n \in \{0,1,2,\ldots,N-1\}$, and N is the number of loudspeakers that are involved in the WFS system.

$Q_{R,n}$ This signal is obtained by delaying a computed WFS signal $\widehat{\tau}_{mn}$ samples, where $\widehat{\tau}_{mn}$ corresponds to the round of τ_{mn} to the nearest integer.

$$\begin{aligned} q_n &= a_{mn}\sin(2\pi\frac{f}{f_s}k), \\ Q_{R,n} &= q_n * \delta[k-\widehat{\tau}_{mn}]. \end{aligned} \tag{20}$$

$Q_{fd,n}$ This signal is obtained from the fractional delay filter. To this end, a WFS signal must be obtained by using $\widetilde{\tau}_{mn}$, where $\widetilde{\tau}_{mn}$ corresponds to the nearest integer below. Afterwards, the obtained signal is filtered by the fractional delay filter h_{fd}

$$\begin{aligned} q_n &= a_{mn}\sin(2\pi\frac{f}{f_s}k), \\ Q_{R,n} &= q_n * \delta[k-\widetilde{\tau}_{mn}], \\ Q_{fd,n} &= Q_{R,n} * h_{fd}. \end{aligned} \tag{21}$$

The generation of the signal of $Q_{fd,n}$ depends on the fractional delay filter: linear $Q_{fd_L,n}$, cubic $Q_{fd_C,n}$, or truncated Lagrange $Q_{fd_T,n}$. In case of the truncated Lagrange, the fractional delay filters have a 10-th order truncated from the 30-th order prototype.

As can be appreciated, filter h from (3) has not been taken into account in all the previous equations, since it has the same influence in the computation of the three different signals. Finally, we define a trajectory, in which a virtual sound source always uses the same 24 loudspeakers that are located in one of the horizontal sides of Fig.2. The 24 driving signals are captured in order to be afterwards analyzed. To compare the three different groups of 24 signals, we compute the energy of the difference sample-by-sample of $Q_{R,n}$ and $Q_{fd,n}$ respect to $Q_{I,n}$ ($E_{RI,n}$, $E_{LI,n}$, $E_{CI,n}$, $E_{TI,n}$). The computation of energy $E_{RI,n}$ is shown in equation (22), which fits also with $E_{LI,n}$, by changing $Q_{R,n}$ for $Q_{fd_L,n}$. The same occurs by combining the energies $E_{CI,n}$ and $E_{TI,n}$ with $Q_{fd_C,n}$ and $Q_{fd_T,n}$, respectively.

$$E_{RI,n} = \sum_{k=0}^{3f_s} (Q_{R,n}[k] - Q_{I,n}[k])^2. \tag{22}$$

This gives us 24 values that correspond to the energy for each one of 24 driving signal, i.e. $n \in \{0, 1, 2, \ldots, 23\}$. To obtain a global metric, we compute finally the mean of these 24 values and obtain: E_{RI}, E_{LI}, E_{CI} and E_{TI}. The computation of energy E_{RI} is shown in equation (23) which fits also with E_{LI}, E_{CI} and E_{TI}, by changing $E_{RI,n}$ for $E_{LI,n}$, $E_{CI,n}$, and $E_{TI,n}$, respectively.

$$E_{RI} = \frac{1}{24} \sum_{n=0}^{24} E_{RI,n}. \tag{23}$$

The measures of the energies $\{E_{RI}, E_{LI}, E_{CI}, E_{TI}\}$ help us to evaluate the discontinuities that are produced during the synthesis of the different driving signals $\{Q_{R,n}, Q_{fd_L,n}, Q_{fd_C,n}, Q_{fd_T,n}\}$. Greater values of energy indicate that the synthesized signal presents high quantity of distortion. Comparing the four energy values, we aim to assess which configuration resembles to the reference signal $Q_{I,n}$.

In order to cover a wide range of systems, we compute the previous parameters for different tonal signals at sample frequency of $f_s = 44100$ samples per second. For low frequency tones, the values of E_{RI}, E_{LI}, E_{CI} and E_{TI} are quite similar, since the wave length is quite large. As the frequency tones increase, the differences among E_{RI}, E_{LI}, E_{CI} and E_{TI} start to be more significant. Table II shows the value of the energies for different trajectory resolutions d_{AB} for the extreme case of f=15 kHz. Note that higher order of fractional filters has a greater influence in the driving signal. In fact, E_{RI}, E_{LI} and E_{CI} present large-scale values with respect to E_{TI} for all the distances d_{AB}. This means that the use of fractional delay filters from truncated Lagrange interpolation reduces the distortion in those signals that own a wide range of frequencies.

Thus, for implementing our WFS system, we use fractional delay filters from truncated Lagrange interpolation for rendering virtual sound sources movements.

VI. Implementation of the WFS processing on the GPU

Once the positions of each virtual sound source has been selected and the input-data buffers $\mathbf{s}_{\mathrm{buff}_m}$ are received at GPU, the processing to carry out in our WFS system is summarized in Algorithm 1:

TABLE II

PARAMETERS E_{RI}, E_{LI}, E_{CI} AND E_{TI} FOR F=15 KHZ, AND DIFFERENT DISTANCES d_{AB}

d_{AB}	f=15 kHz			
	E_{RI}	E_{LI}	E_{CI}	E_{TI}
0.0001	8218.8825	3229.3721	1304.5125	**4.0037**
0.0010	8221.2549	3232.0923	1316.4483	**40.1581**
0.0025	8234.6420	3236.7583	1336.0903	**98.9571**
0.0050	8195.2193	3243.6432	1363.6179	**185.6639**
0.0010	8349.2533	3324.3441	1426.7089	**286.9322**

Algorithm 1 WFS system with Room Compensation using Fractional Delays Filters.

Input: M, θ_{mn}, r_{mn}, $\mathbf{s}_{\text{buff}_m}$, $\widetilde{f}_{jn}$

Output: $\mathbf{y}_{\text{buff}_n}$

/*----- Driving Signals of WFS ----*/
for $n = 0, \ldots, N-1$ **do** ▷*Computation of Amplitudes and Delays*
  $\mathbf{q}_{\text{buff}_n}$ = 0; ▷*Initialization of buffers to Zeros*
  for $m = 0, \ldots, M-1$ **do** ▷*Combination Loudspeaker-Sound Source*
    $a_{mn} = \frac{K}{\sqrt{r_{mn}}} \cos(\theta_{mn})$. ▷$r_{mn} = \mid \mathbf{x}_m - \mathbf{x}_n \mid$;
    $\tau_{mn} = \frac{r_{mn}}{c} f_s$. ▷*see Fig (3) for* θ_{mn}.
    h_{fd} = Compute_Fractional_Delay_Filters(τ_{mn}).
    $\mathbf{q}^m_{\text{buff}_n}$ = $a_{mn} \cdot (\mathbf{s}_{\text{buff}_m} * h_{fd})$.
    $\mathbf{q}_{\text{buff}_n}$ = $\mathbf{q}_{\text{buff}_n} + \mathbf{q}^m_{\text{buff}_n}$.
  end for
end for
/*----- Room Compensation filtering ----*/
for $n = 0, \ldots, N-1$ **do**
  $\mathbf{y}_{\text{buff}_n}$ = 0; ▷*Initialization of output-buffers to Zeros*
  for $j = 0, \ldots, M-1$ **do**
    $\mathbf{y}_{\text{buff}_n}$ = $\mathbf{y}_{\text{buff}_n}$ + $(\mathbf{q}_{\text{buff}_j} * \widetilde{f}_{jn})$. ▷*Filter* h *is included. See Equation (19)*
  end for
end for

The following subsections details the GPU implementation of each processing block: Driving Signals of WFS, and Room Compensation filtering.

A. Driving Signals of WFS

We launch the following CUDA kernels in order to compute from step 2 to step 11 in algorithm 1.

Kernel 1 launches NM threads and computes steps 5 and 6. The input of the kernels are the coordinates of the virtual sound sources and the positions of all the loudspeakers. Each thread is devoted to compute a τ_{mn} and a a_{mn}.

Kernel 2 This kernel computes the fractional delay filters (step 7). As it was designed in Section V, each filter is composed of ten coefficients and, as there is a filter per sound source and loudspeaker, $10NM$ threads are launched. Each thread is devoted to compute a coefficient of one of the filters.

Kernel 3 This kernel computes step 8. To this end, a tridimensional matrix composed of the audio buffers $\mathbf{q}^m_{\mathrm{buff}_n}$ is configured. As every filter is composed of 10 coefficients, convolution is carried out in time domain (each thread performs 10 multiplications and 10 sums). The number of rows of this matrix corresponds to the number of loudspeakers N, the number of columns is $2L$ (number of samples per buffer, see section II-B), and the third dimension corresponds to the number of sources M in the WFS system. In total, $2LNM$ threads are launched. The configuration of this matrix plays an important role in order to apply efficiently next steps of the implementation. It is important to remark that the threads that use the audio samples access to *global-memory* in a coalesced way. Moreover, in case the Nvidia device has a compute capability of 3.5, the cuda compiler is forced to use the *read-only* memory path for loading audio samples from the the *global-memory*, since this memory has a high bandwidth. Last feature to configure in the cuda programming was the L1 cache that was set to 48 kB in order to cache memory loads from *global-memory*. These three actions reduce the possible memory conflicts when multiple threads access to *global-memory* concurrently.

Kernel 4 is devoted to compact all the samples per loudspeaker. To this end, $2LN$ threads are launched and each thread performs M sums.

B. Room Compensation

Previous work [42] dealt with filters that had the same length as the sample buffers. In fact, the length that was previously considered was $L = 512$. However, the filters that carry out the Room Compensation has usually a large number of coefficients [55]. In case of using audio buffers with the same length, the latency of the system would increase substantially. This would imply also that the movements of the virtual sound sources were slower.

For this new implementation we are going to add also the development carried out in [29]. That work presented a GPU-based implementation that aims to efficiently filtering audio buffers that have a size much

TABLE III

OPERATIONS THAT ARE REQUIRED BY THE ROOM COMPENSATION BLOCK WHEN IT IS APPLIED UNIFORMLY PARTITIONED FFT-BASED CONVOLUTION.

Stage	Number of computations	Operation description
Room Compensation	N	FFT of size $2L$
	PN^2	1 Element-wise Multiplication of size $2L$ in (12)
	PN^2	1 Element-wise Sum with the elements that were generated by the previous buffer
	N	N Element-wise Sum of size $2L$ in (12)
	N^2	Set to Zero $2L$ elements
	N	Inverse FFT of size $2L$

smaller than the size of the filters in a partitioned way. This is performed by using uniformly-partitioned FFT-based convolution in the frequency-domain, which is described by Wefers et al in [56].

Table III summarizes the operations that are executed by the Room Compensation block when it is applied uniformly partitioned massive filtering, where P denotes the number of partitions of size $2L$

The N^2 filtering operations are carried out in the frequency domain [57]. That means that convolution is converted in an element-wise multiplication of two vectors of size $2L$ composed of complex values. To this end, N $2L$-FFT transformations are performed by using the Nvidia library CUFFT [44]. Afterwards, two kernels more are executed. Both kernels are devoted to execute the massive filtering of $\mathbf{q}_{\mathrm{buff}_n}$ with the filters $\widetilde{f}_{jn}$ (step 16 of Algorithm 1), which arose from the convolution of the room compensation filters f_{jn} with the filter h, as shown in (13). These two GPU kernels are described in [29] (pseudo cuda codes of these two kernels are labeled as CUDA kernel 6 and CUDA kernel 7 at the dissertation [58]).

Finally, N inverse $2L$-FFT transformations are performed by using the Nvidia library CUFFT. The implementation ends by transferring the N output-data buffers $\mathbf{y}_{\mathrm{buff}_n}$ to the CPU in order to be rendered by the loudspeakers.

VII. RESULTS

We perform our WFS system on the Nvidia device K20Xm card (see Table IV for main characteristics), which is one of the latest Kepler-based family of GPUs. The proposed WFS system is analyzed attending to: different sizes of audio buffers L and different sizes of room compensation filters.

In order to perform the different implementations, we set the audio card to provide blocks of samples of size $L \in \{32, 64, 128, 256, 512, 1024\}$ at a sample frequency $f_s = 44.1$ kHz. This means that the

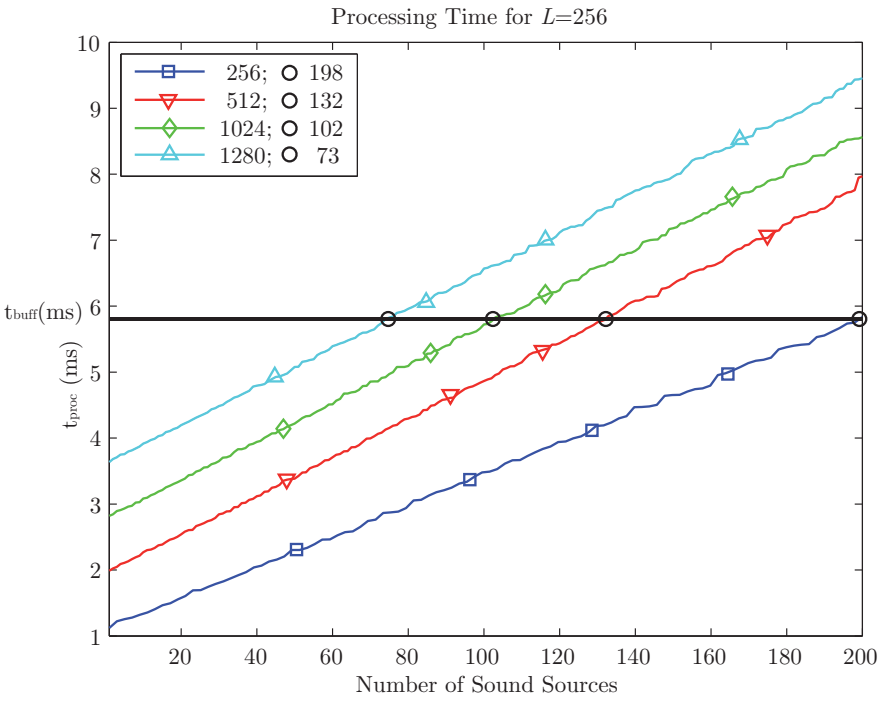

Fig. 6. Performance of our WFS system using a buffer size of $L = 256$ samples for different sizes of Room Compensation Filters: 256, 512, 1024 and 1280.

TABLE IV

CHARACTERISTICS OF THE NVIDIA DEVICE K20XM.

Nvidia Device	Tesla K20Xm
Architecture	Kepler GK110
Compute Capability	3.5
Number of SM	14
Number of cores/SM	192
Total number of cores	2688
Read-only Cache	Yes

times t_{buff} of the system are 0.72 ms, 1.45 ms, 2.90 ms, 5.80 ms, 11.61 ms, and 23.22 ms, for each one of the buffer sizes. We perform our WFS system by increasing the number of sources in the system M gradually and by measuring each time t_{proc} for the different described environments. As commented in Section II-B, our WFS system allows to work under real-time conditions as long as $t_{\text{proc}} < t_{\text{buff}}$.

Figure 6 shows the results when a system is executed using a buffer size of $L = 256$ samples. Computational performance has been assessed by assuming Room compensation filters $\widetilde{f}_{jn}$ with sizes

256, 512, 1024, and 1280 coefficients. We execute the system by increasing M (number of sound sources) gradually and by measuring each time t_{proc}. Note that the maximum number of filters that can be executed in real time are marked with a circle $\circ$ in Fig. 6, and their values are shown in the legend of the figure for all cases.

Table V completes the results shown in Fig. 6 for different buffer sizes L and different sizes of Room Compensation Filters. Column 3 shows the number of partitions P of size $2L$ that is produced in each one of the 96 $\times$ 96 filters $\widetilde{f}_{jn}$ that compose our inverse filter bank (Fig. 4). Column 5 indicates the maximum number of sound sources that can be rendered in real time in our system. The time t_{proc} employed by the GPU in computing that number of sound sources is shown in column 6. As a reference, column 4 indicates the time employed by the GPU in computing one sound source.

Performing our 96-loudspeaker system attending to the previous features allows us to establish: 1) specific speeds that can be applied to the virtual sound sources; 2) maximum number of sound sources that can be rendered in real time; and 3) maximum number of coefficients that can have the filters $\widetilde{f}_{jn}$.

The speed of the sound source is given by the ratio between the distance d_{AB} and the time inverted on changing from position A to position B, i.e Lf_s seconds. Thus, the speed of the virtual sound source is mainly determined by the size of the buffer L, since the sample frequency is fixed and d_{AB} is given by a required trajectory resolution in a WFS system. Table VI shows different speeds that can be achieved for different combinations of sizes of buffer L with different resolutions d_{AB} in the WFS system.

Table V and Table VI allow to obtain the performance of a 96-loudspeakers WFS system attending to: 1) a specific resolution for the movements of the virtual sound sources, and 2) size of Room Compensation Filters. As an example, let us suppose that we want to design a WFS system that produce sound source movements between positions with a required resolution of 0.007 m and with a speed of around 0.6 m/s. Then, looking at Table VI, we can observe that both specifications are achieved by using audio buffers of size $L = 512$. Afterwards, we must take into account how reverberant is the room where the WFS system is located, and taking this into account, we design the room compensation filters f_{jn} considering that the maximum length that they can have is $3072 - l_h + 1$ (l_h is the length of filter h from (13)). To this specifications, the maximum number of sound sources that the WFS system is able to render in real time is 21 sound sources.

VIII. Conclusion

The use of the GPU in large-scale audio systems is getting more widespread. One of the audio systems that deals with multiple sound sources and loudspeakers is a Wave Field Synthesis system with Room

TABLE V

MAXIMUM PERFORMANCE THAT CAN GIVE OUR WFS SYSTEM FOR DIFFERENT BUFFER SIZES L AND DIFFERENT SIZES OF ROOM COMPENSATION FILTERS $\widetilde{f}_{jn}$.

L	t_{buff} (ms)	P	Size of $\widetilde{f}_{jn}$	t_{proc} (ms) 1 source	Max. Sources Real Time	t_{proc} (ms) Real Time
32	0.725	1	32	0.270	47	0.721
		2	64	0.338	36	0.716
		3	96	0.439	27	0.710
		4	128	0.547	18	0.713
		5	160	0.646	7	0,706
		6	192	0,753	0	0
		7	224	0.865	0	0
64	1.451	1	64	0.391	92	1.433
		2	128	0.579	68	1.449
		3	192	0.780	51	1.449
		4	256	0.999	36	1.442
		5	320	1.187	20	1.436
		6	384	1.403	5	1.446
		7	448	1.605	0	0
128	2.902	1	128	0.648	146	2.881
		2	256	1.052	103	2.893
		3	384	1.466	79	2.892
		4	512	1.872	56	2.894
		5	640	2.285	34	2.892
		6	768	2.701	12	2.891
		7	896	3.117	0	0
256	5.805	1	256	1.186	198	5.752
		2	512	1.983	132	5.793
		3	768	2.821	102	5.770
		4	1024	3.634	73	5.744
		5	1280	4.457	45	5.750
		6	1536	5.303	18	5.791
		7	1792	6.115	0	0
512	11.610	1	512	2.267	232	11.606
		2	1024	3.894	147	11.579
		3	1536	5.542	117	11.583
		4	2048	7.215	85	11.569
		5	2560	8.861	53	11.527
		6	3072	10.509	21	11.517
		7	3584	12.147	0	0
1024	23.22	1	1024	4.343	262	23.175
		2	2048	7.650	162	23.062
		3	3072	10.939	130	23.198
		4	4096	14.234	94	23.092
		5	5120	17.618	60	23.199
		6	6144	20.880	25	23.121
		7	7168	24.153	0	0

TABLE VI

SPEED OF A VIRTUAL SOUND SOURCE IN A WFS SYSTEM FOR DIFFERENT COMBINATIONS OF SIZES OF BUFFER L AND OF RESOLUTIONS d_{AB}.

Speed of a virtual sound source (m/s)						
d_{AB} (m)	Sizes of buffers L (samples)					
	32	64	128	256	512	1024
0.0001	0.137	0.068	0.034	0.017	0.008	0.004
0.001	1.378	0.689	0.344	0.172	0.086	0.043
0.0025	3.445	1.722	0.861	0.430	0.215	0.107
0.005	6.890	3.445	1.722	0.861	0.430	0.215
0.007	9.646	4.823	2.411	1.205	0.602	0.301

Compensation. In this article, we have presented a detailed study about the performance that a GPU achieves in our 96-loudspeaker WFS system taking into account three different aspects: specific resolutions in the movements of the sound sources, length of the room compensation filters, and maximum number of sound sources to be rendered under real-time conditions.

To overcome the limitations on the accuracy of the sound source positions, we have dealt with fractional delay filters. Different kinds of fractional filters have been studied in this article in order to select the proper filters to delay a signal a number of samples that is not an integer value.

On the other hand, we have improved our previous GPU-based implementation by filtering audio buffers shorter than the length of room compensation filters. This allows to design filters with a large number of coefficients and thus, to obtain better room compensations.

For implementing the system efficiently, we have convolved the WFS pre-equalization filter h with the room compensation filters f_{jn} before starting the real-time processing.

Finally, the presented implementation exploits the resources of GPUs with Kepler architecture, which can be currently found in new generation mobile devices such as Google's Nexus 9 tablet. Thus, the proposed implementation can be successfully run on embedded mobile devices.

ACKNOWLEDGMENT

This work has been partially funded by Spanish Ministerio de Economía y Competitividad (TEC2012-37945-C02-02), Generalitat Valenciana PROMETEO 2009/2013, and Universitat Politècnica de València through Programa de Apoyo a la Investigación y Desarrollo (PAID-05-11 and PAID-05-12).

REFERENCES

[1] A. Berkhout, "A holographic approach to acoustic control," *J. of the Audio Engineering Society*, vol. 36, pp. 2764–2778, May 1988.

[2] A. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by Wave Field Synthesis," *J. Acoustic. Soc. Amer*, vol. 93, pp. 2764–2778, May 1993.

[3] P. Vogel, "Application of Wave Field Synthesis in Room Acoustics," Ph.D. dissertation, Delft University of Technology, 1993.

[4] E. Start, "Direct sound enhancement by Wave Field Synthesis," Ph.D. dissertation, Delft University of Technology, 1997.

[5] E. Verheijen, "Sound reproduction by Wave Field Synthesis," Ph.D. dissertation, Delft University of Technology, 1997.

[6] J.-J. Sonke, "Variable acoustics by Wave Field Synthesis," Ph.D. dissertation, Delft University of Technology, 2000.

[7] E. Hulsebos, "Auralization using Wave Field Synthesis," Ph.D. dissertation, Delft University of Technology, 2004.

[8] E. Hulsebos, D. de Vries, and E. Bourdillat, "Improved microphone array configurations for auralization of sound fields by Wave-Field Synthesis," *Journal of the Audio Engineering Society*, vol. 50, no. 10, pp. 779–790, 2002.

[9] S. Spors, H. Buchner, and R. Rabenstein, "Efficient active listening room compensation for Wave Field Synthesis," in *Proceedings of the 116th AES Convention*, Berlin,Germany, May 2004.

[10] J. Lopez, A. Gonzalez, and L. Fuster, "Room compensation in Wave Field Synthesis by means of multichannel inversion," in *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, oct. 2005, pp. 146 – 149.

[11] T. Laakso, V. Välimäki, M. Karjalainen, and U. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, 1996.

[12] M. Karjalainen, T. Paatero, J. Pakarinen, and V. Välimäki, "Special digital filters for audio reproduction," in *Proc. of the 32nd AES Conference*, Hillerod, Denmark, September 2007.

[13] J. Dattorro, "Effect design?part 2: Delay-line modulation and chorus." *J. Audio Eng. Soc*, vol. 45, no. 10, pp. 764–788, 1997.

[14] U. Zölzer, "Dafx - digital audio effects (second edition)," Edited by Udo Zölzer, 2011.

[15] V. Välimäki, "Discrete-time modeling of acoustic tubes using fractional delay filters," Ph.D. dissertation, Helsinki University of Technology, 1995.

[16] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Reports on Progress in Physics*, vol. 69, no. 1, pp. 1–78, 2006.

[17] A. Franck, A. Gräfe, T. Korn, and M. Strauss, "Reproduction of moving sound sources by Wave Field Synthesis: An analysis of artifacts." in *Proc. of the 32nd AES Conference*, Hillerod, Denmark, September 2007.

[18] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics," in *Proc. of the Int. Conf. Digital Audio Effects*, Graz, Austria, September 2010.

[19] A. Southern, D. Murphy, G. Campos, and P. Dias, "Finite difference room acoustic modelling on a General Purpose Graphics Processing Unit," in *Proc. of the 128th AES Convention*, London, United Kingdom, May 2010.

[20] C. J. Webb and S. Bilbao, "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," in *Proc. IEEE International Conference on Acoustics, Speech and Signal (ICASSP)*, Prague, Czech Republic, May 2011.

[21] B. Hamilton and C. J. Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, September 2013.

[22] L. Savioja, V. Välimäki, and J. O. Smith, "Real-time additive synthesis with one million sinusoids using a GPU," in *Proc. of the 128th AES Convention*, London, United Kingdom, May 2010.

[23] ——, "Audio Signal Processing using Graphics Processing Units," *J. Audio Eng. Soc*, vol. 59, no. 1-2, pp. 3–19, 2011.

[24] S. Bilbao and C. J. Webb, "Physical modeling of timpani drums in 3D on GPGPUs," *J. Audio Eng. Soc*, vol. 61, no. 10, pp. 737–748, 2013.

[25] R. Bradford, J. Ffitch, and R. Dobson, "Real-time sliding phase vocoder using a commodity GPU," in *Proc. of ICMC 2011*, University of Huddersfield, United Kingdom, August 2011.

[26] J. Lorente, G. Piñero, A. Vidal, J. Belloch, and A. Gonzalez, "Parallel implementations of beamforming design and filtering for microphone array applications," in *Proc. of EUSIPCO 2011*, Barcelona, Spain, August 2011.

[27] N. Tsingos, W. Jiang, and I. Williams, "Using Programmable Graphics Hardware for Acoustics and Audio Rendering," *J. Audio Eng. Soc*, vol. 59, no. 9, pp. 628–646, 2011.

[28] 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.

[29] J. A. Belloch, A. Gonzalez, F. Martinez-Zaldivar, and A. M. Vidal, "Multichannel massive audio processing for a generalized crosstalk cancellation and equalization application using GPUs," *Integrated Computer-Aided Engineering*, vol. 20, no. 2, pp. 169–182, 2013.

[30] 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.

[31] J. A. Belloch, J. Parker, L. Savioja, A. Gonzalez, and V. Välimäki, "Dynamic range reduction of audio signals using multiple allpass filters on a GPU accelerator," in *Proceedings of 22nd European Signal Processing conference, (EUSIPCO)*, Lisbon, Portugal, September 2014.

[32] M. Schneider, F. Schuh, and W. Kellermann, "The Generalized Frequency-Domain Adaptive Filtering Algorithm Implemented on a GPU for Large-Scale Multichannel Acoustic Echo Cancellation," *Speech Communication; 10. ITG Symposium; Proceedings of*, pp. 1 –4, sept. 2012.

[33] J. Lorente, A. Gonzalez, M. Ferrer, J. A. Belloch, M. De Diego, G. Piñero, and A. M. Vidal, "Active noise conrol using Graphics Processing Units," in *Proc of the International Congress on Sound and Vibration*, Vilnius, Lithuania, July 2012.

[34] J. Lorente, M. Ferrer, M. De Diego, J. A. Belloch, and A. Gonzalez, "GPU implementation of a frequency-domain modified filtered-X LMS algorithm for multichannel local active noise control," in *Proc. of the 52nd AES Conference*, Guildford, United Kingdom, September 2013.

[35] L. Romoli, P. Peretti, S. Cecchi, L. Palestini, and F. Piazza, "Real-time implementation of Wave Field Synthesis for sound reproduction systems," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, 30 2008-dec. 3 2008, pp. 430 –433.

[36] D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "A minimalistic architecture for reconfigurable wfs-based immersive-audio," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, dec. 2010, pp. 1 –6.

[37] ——, "Multi-core platforms for beamforming and Wave Field Synthesis," *IEEE Transactions on multimedia*, vol. 3, no. 2, pp. 235–245, April 2011.

[38] R. Ranjan and W. S. Gan, "Fast and efficient real-time GPU based implementation of Wave Field Synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014.

[39] K20, "NVIDIA Kepler Architecture,"

http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf, 2014, (accessed 2014 August 27).

[40] Jetson, "Mobile GPU: Jetson," http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf, 2015, (accessed 2014 November 22).

[41] Nexus, "Google's nexus 9," http://blogs.nvidia.com/blog/2014/10/17/nvidia-tegra-k1-google-nexus-9/, 2015, (accessed 2015 January 11).

[42] J. Belloch, M. Ferrer, A. Gonzalez, J. Lorente, and A. Vidal, "GPU-based WFS Systems with Mobile Virtual Sound Sources and Room Compensation," in *Proc. of the 52nd AES Conference*, Guildford, United Kingdom, September 2013.

[43] M. Flynn, "Some computer organizations and their effectiveness," *IEEE Transactions on Computers*, vol. 21, pp. 948–960, 1972.

[44] "Nvidia CUDA Developer Zone," https://developer.nvidia.com/cuda-zone, (accessed 2014 April 10).

[45] "Audio and Communications Signal Processing Group at Universitat Politecnica de Valencia," *http://www.gtac.upv.es*.

[46] M. M. Boone, E. N. G. Verheijen, and P. F. Van Tol, "Spatial Sound-Field Reproduction by Wave-Field Synthesis," *J. Audio Eng. Soc*, vol. 43, no. 12, pp. 1003–1012, 1995.

[47] S. Spors, A. Kuntz, and R. Rabenstein, "An Approach to Listening Room Compensation with Wave Field Synthesis," in *Proc. of the 24th AES Conference*, Banff, Canada, May 2003.

[48] S. Spors and J. Ahrens, "Analysis and Improvement of Pre-equalization in 2.5-Dimensional Wave Field Synthesis," in *Proceedings of the 128th AES Convention*, London, UK, May 2010.

[49] L. Fuster, J. J. Lopez, A. Gonzalez, and P. Faus, "Time and frequency domain room compensation applied to Wave Field Synthesis," in *Proc. Conference on Digital Audio Effects (DAFx-05)*, Madrid, Spain, September 2005.

[50] O. Kirkeby, P. Nelson, H. Hamada, and F. Orduna-Bustamante, "Fast deconvolution of multichannel systems using regularization," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, pp. 189 –194, Mar 1998.

[51] M. Miyoshi and Y. Kaneda, "Inverse filtering of room acoustics," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 2, pp. 145 –152, Feb 1988.

[52] A. Franck, "Efficient algorithms for arbitrary sample rate conversion with application to Wave Field Synthesis," Ph.D. dissertation, Technical University of Ilmenau, 2012.

[53] V. Välimäki and A. Haghparast, "Fractional Delay Filter Design Based on Truncated Lagrange Interpolation," *IEEE Signal Processing Letters*, vol. 14, no. 11, p. 816?819, November 2007.

[54] E. Hermanowicz, "Explicit formulas for weighting coefficients of maximally flat tunable fir delayers," *Electron Letters*, vol. 28, no. 2, pp. 1936–1937, September 1992.

[55] H. Kuttruff, *Room acoustics*, S. Press, Ed. Abingdon, Oxford, UK: Taylor & Francis, October 2000, 368 pages.

[56] F. Wefers and J. Berg, "Optimal filter partitions for real-time FIR filtering using uniformly-partitioned FFT-based convolution in the frequency-domain," in *Proc. of the 14th Conference on Digital Audio Effects*, Paris, France, September 2011.

[57] A. V. Oppenheim, A. S. Willsky, and S. Hamid, "Signals and systems," ser. Processing series. Prentice Hall, 1997.

[58] J. A. Belloch, "Performance Improvement of Multichannel Audio by Graphics Processing Units," Ph.D. dissertation, Universitat Politecnica de Valencia, 2014.