KU LEUVEN

STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Authors: Hannes Rosseel and Toon van Waterschoot Title: Accelerated Interactive Auralization of Highly Reverberant Spaces using Graphics Hardware Journal: Journal of the Audio Engineering Society Pages: 1–8
Archived version	■ Submitted version ☐ Accepted (Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher) ☐ Published ☐ Published (Open Access)
Acknowledgements	In this manuscript, the following funding is acknowledged: ■ EU H2020 ☐ EU Horizon Europe ■ FWO ■ KULeuven Internal Funds ☐ VLAIO ■ AI ☐ Other
Published version	
Abstract	Interactive acoustic auralization allows users to explore virtual acoustic environments in real-time, enabling the acoustic recreation of concert hall or Historical Worship Spaces (HWS) that are either no longer accessible, acoustically altered, or impractical to visit. Interactive acoustic synthesis requires real-time convolution of input signals with a set of synthesis filters that model the space-time acoustic response of the space. The acoustics in concert halls and HWS are both characterized by a long reverberation time, resulting in synthesis filters containing many filter taps. As a result, the convolution process can be computationally demanding, introducing significant latency that limits the real-time interactivity of the auralization system. In this paper, the implementation of a real-time multichannel loudspeaker-based auralization system is presented. This system is capable of synthesizing the acoustics of highly reverberant spaces in real-time using GPU-acceleration. A comparison between traditional CPU-based convolution and GPU-accelerated convolution is presented, showing that the latter can achieve real-time performance with significantly lower latency. Additionally, the system integrates acoustic synthesis with acoustic feedback cancellation on the GPU, creating a unified loudspeaker-based auralization framework that minimizes processing latency.
Author contact	hannes.rosseel@esat.kuleuven.be

Accelerated Interactive Auralization of Highly Reverberant Spaces using Graphics Hardware

Hannes Rosseel and Toon van Waterschoot
KU Leuven, Dept. of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Leuven, Belgium
{firstname.lastname}@esat.kuleuven.be

***Abstract*—Interactive acoustic auralization allows users to explore virtual acoustic environments in real-time, enabling the acoustic recreation of concert hall or Historical Worship Spaces (HWS) that are either no longer accessible, acoustically altered, or impractical to visit. Interactive acoustic synthesis requires real-time convolution of input signals with a set of synthesis filters that model the space-time acoustic response of the space. The acoustics in concert halls and HWS are both characterized by a long reverberation time, resulting in synthesis filters containing many filter taps. As a result, the convolution process can be computationally demanding, introducing significant latency that limits the real-time interactivity of the auralization system. In this paper, the implementation of a real-time multichannel loudspeaker-based auralization system is presented. This system is capable of synthesizing the acoustics of highly reverberant spaces in real-time using Graphics Processing Unit (GPU)-acceleration. A comparison between traditional Central Processing Unit (CPU)-based convolution and GPU-accelerated convolution is presented, showing that the latter can achieve real-time performance with significantly lower latency. Additionally, the system integrates acoustic synthesis with acoustic feedback cancellation on the GPU, creating a unified loudspeaker-based auralization framework that minimizes processing latency.**

I. Introduction

Room acoustic auralization has been a topic of interest in audio engineering for several decades [1]–[3]. Auralization refers to the process of recreating an acoustic environment through sound reproduction techniques. This is typically achieved by convolving an audio signal with a set of auralization filters that model the acoustic characteristics of a given space. When the output of this convolution process is played back through loudspeakers or headphones, the listener perceives the sound as if it was recorded in the modeled acoustic environment [3]. Applications of room acoustic auralization span various domains, including architectural acoustics, virtual reality, immersive art installations, and music production, among others [3].

Interactive auralization extends these principles by enabling real-time exploration of the reproduced acoustic environment, allowing users to experience and interact with the various acoustic environments through live audio input. For vocal musicians and researchers of early music, interactive auralization provides a valuable tool to explore the intrinsic connection between the performance of music and the acoustics of the space for which the music was composed [4]–[6]. This is particularly relevant for concert halls and HWS, where interactive auralization facilitates the preservation and study of the unique acoustic properties of these culturally significant locations [7].

An important consideration of interactive auralization systems is latency, as significant delays between user interactions and system responses can degrade the sense of immersion and responsiveness [3], [8]. To ensure real-time performance, auralization systems must perform the acoustic synthesis and any other processing in the sound acquisition-reproduction loop with minimal latency, ensuring that the synthesized sound field is perceived by the users without noticeable delay. The maximum allowable delay between the input audio signal and the resulting synthesized sound field is referred to as the latency budget [3]. For block-based processing, the latency budget is defined as n_x/f_s, where n_x denotes the block size in samples and f_s is the operating sampling frequency of the auralization system. For sample-based processing, the latency budget is defined as $1/f_s$, requiring the system to process each sample within a single sample period.

The modeling of rich and complex acoustics, such as those found in concert halls and HWS, require many filter taps to accurately capture the long reverberation times and spatial characteristics of the space. During real-time auralization, the convolution of input audio with these filters can be computationally expensive, especially in multichannel auralization systems containing many loudspeakers, as each loudspeaker is associated with its own synthesis filter. This makes meeting the desired latency budget, which is typically in the order of a few milliseconds, challenging to achieve using traditional consumer-grade hardware.

Moreover, in loudspeaker-based auralization systems that allow for interactive exploration of virtual acoustic environments, the presence of acoustic feedback can introduce additional challenges. Acoustic feedback can occur when the microphone capturing input audio is positioned near the loudspeakers used to reproduce the synthesized sound field. When the synthesized sound field is picked up by the microphone and reintroduced into the system, a closed feedback loop is formed. This can cause system instability, leading to undesirable artifacts such as howling and coloration of the synthesized sound [9]. To mitigate this issue in real-time interactive auralization systems, where the feedback path between the loudspeakers and the microphone is relatively time-invariant, feedback cancellation algorithms that identify and cancel the feedback signal have been proposed [10]. It should be noted that in these algorithms, it is assumed that the

feedback path between the loudspeakers and the microphones is known a priori and remains relatively time-invariant during the operation of the system.

Traditional implementations of real-time auralization systems are often constrained by the computational complexity of the convolution process, and therefore limited in the number of channels that can be processed in real-time. To address this challenge, GPU acceleration has been proposed as a means to reduce the processing latency for multichannel auralization by executing the convolution operations in parallel [11], [12]. Prior work has demonstrated the effectiveness of GPU acceleration for real-time binaural auralization [13]–[18], multichannel audio systems [19], and Wave Field Synthesis (WFS) auralization [20]–[22]. Notably, Belloch et al. [22] implemented a real-time GPU-based WFS system with room compensation that is capable of auralizing complex sound fields with moving sources. This work demonstrated the feasibility of using GPU acceleration for real-time auralization and highlighted the potential benefits of GPU acceleration for reducing latency and improving computational efficiency in auralization systems. However, the integration of feedback cancellation algorithms with GPU-based convolution for real-time interactive auralization has not been extensively explored in the literature.

This paper aims to bridge this gap by presenting a generalizable framework for real-time multichannel auralization systems that integrates uniform partitioned convolution [11], [23] with real-time feedback cancellation, and utilizes GPU acceleration to minimize processing latency. Unlike previous GPU-based approaches that focus on specific auralization techniques such as WFS [20]–[22], our framework is designed to be flexible and applicable to a wider range of loudspeaker-based auralization scenarios, with virtually no limitation on the length of the synthesis filters and the amount of channels in the reproduction setup. This makes our framework particularly well-suited for modeling complex acoustic environments with long reverberation times, including concert hall and HWS acoustics.

To validate the effectiveness of the proposed approach, we conduct a performance evaluation comparing the performance of the GPU-accelerated implementation to a CPU-based implementation. Our results demonstrate that the proposed framework can achieve low-latency real-time auralization of HWS acoustics with minimal system instability for large filter lengths and channel counts. The results highlight the potential of GPU acceleration for reducing latency and improving computational efficiency in real-time interactive auralization systems, while maintaining system stability in the presence of acoustic feedback. Additionally, the source code for our framework is made publicly available to support further research and development in interactive auralization.

The paper is organized as follows. In Sec. II, the mathematical notation used throughout the paper is introduced. Sec. III provides an overview of the proposed system implementation, including a detailed description of fast convolution methods. In Sec. IV, the uniform partitioned convolution algorithm used in the proposed framework for real-time auralization is presented. Sec. V details the Python implementation of the partitioned convolution algorithm and the real-time auralization system. In Sec. VI, the results of the performance evaluation of the proposed framework is presented, comparing the GPU-based implementation with a CPU-based implementation. Finally, Sec. VII concludes the paper and discusses future research directions.

II. Mathematical notation

In this paper, scalar quantities are denoted by italic letters (e.g., a or A), while vectors are represented by bold lowercase letters (e.g., $\mathbf{a}$). Matrices are denoted by bold uppercase letters (e.g., $\mathbf{A}$). Quantities that vary over time are denoted by an argument in parentheses (e.g., $a(t)$ or $\mathbf{a}(t)$).

III. Interactive multichannel auralization

Interactive multichannel auralization systems aim to reproduce the acoustics of a virtual or modelled environment around a listener in real-time. In this paper, we aim to develop an interactive multichannel auralization system that can be used as a rehearsal space for vocal performers and musicians. The system should be able to reproduce complex and rich acoustic environments, such as those of HWS, in real-time, allowing the performers to hear themselves and other performers as they would in a real space. Since the setup of the system is intended to be operated by vocal performers and musicians, which are not necessarily experts in acoustics or signal processing, the system should be straightforward to set up without the need for installing additional equipment. Therefore, in this paper, we focus on static loudspeaker-based interactive auralization systems, as opposed to headphone-based binaural auralization systems.

A. System overview

The block diagram of an interactive loudspeaker-based multichannel auralization system is shown in Fig. 1. The system consists of a reproduction area that is surrounded by L loudspeakers and P microphones, which are used to reproduce the acoustics of the virtual environment and capture the input signals, respectively. The input signals are first conditioned using a pre-processing stage $\mathbf{G}$, a $P \times Q$ matrix of operators, which can include equalization, compression, or weighting for different input channels. The conditioned input signals in the $Q \times 1$ vector $\tilde{\mathbf{m}}(t)$ are then filtered with a set of L synthesis filters, after which the resulting output signals in the $L \times 1$ $\mathbf{l}(t)$ are played back by the loudspeakers. The output signals are described by

$$\mathbf{l}(t) = \tilde{\mathbf{m}}(t) * \mathbf{H}_{\text{aur}}, \tag{1}$$

where $\mathbf{H}_{\text{aur}}$ is an $L \times Q$ matrix of linear filter operators, and $(*)$ denotes the convolution operator between one signal in $\tilde{\mathbf{m}}(t)$ and one filter in $\mathbf{H}_{\text{aur}}$. The synthesis filters $\mathbf{H}_{\text{aur}}$ are designed to render the acoustics of the virtual or modeled environment inside the reproduction area. These filters are the result of applying Sound Field Analysis (SFA) and Sound Field Synthesis (SFS) methods to a measured or modeled Room Impulse Response (RIR) [3], [24]. For an overview of

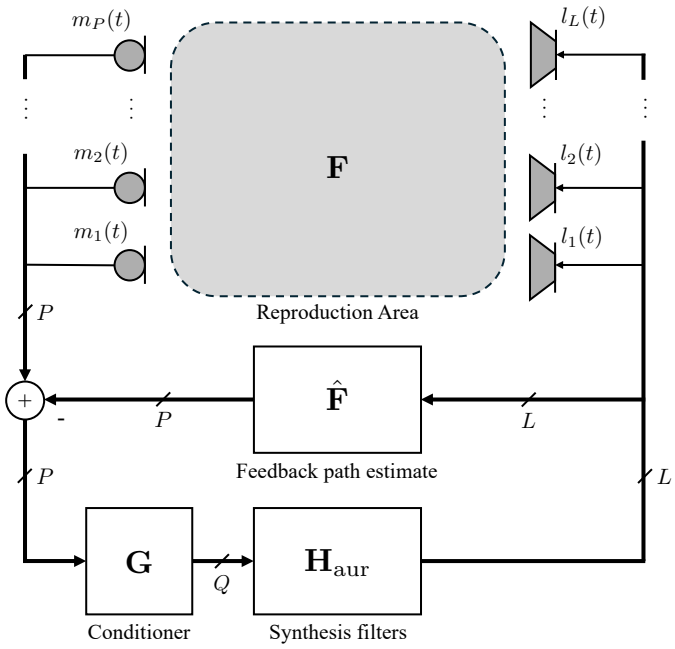

Fig. 1. Block diagram of an interactive multichannel auralization system containing L loudspeakers and P microphones. The system consists of a pre-processing stage, where the microphone signals are processed by a pre-processor function $\mathbf{G}$, a synthesis stage with synthesis filters $\mathbf{H}_{\text{aur}}$, and an acoustic feedback cancellation stage which subtracts the estimated feedback signals from the microphone input. The feedback path for each loudspeaker-microphone pair is estimated a-priori and stored in the estimated feedback path matrix $\hat{\mathbf{F}}$.

SFA and SFS methods for auralization, we refer the reader to [24].

In loudspeaker-based auralization systems, each loudspeaker and microphone pair has an acoustic feedback path $\mathbf{f}_{lp}$, which models the acoustic path from loudspeaker l to microphone p. The collection of all acoustic feedback paths in the auralization system can be represented by the acoustic feedback matrix $\mathbf{F}$, which is a $P \times L$ matrix of linear filter operators. To prevent acoustic feedback in the system, acoustic feedback cancellation can be achieved by first estimating the acoustic feedback path matrix $\hat{\mathbf{F}}$, which models the acoustic path between all microphone and loudspeaker pairs. Subsequently, the feedback signals can be obtained by filtering the output signal $\mathbf{l}(t)$ with the estimated acoustic feedback path matrix $\hat{\mathbf{F}}$. The acoustic feedback is then canceled by subtracting the feedback signals from the microphone signals. Various methods for estimating the acoustic feedback path matrix $\hat{\mathbf{F}}$ have been proposed in the literature [9], [10].

To allow interactive auralization of virtual or modeled acoustics in real-time, the system should be able to perform two main operations under real-time constraints: (1) the convolution of the input signals with the synthesis filters $\mathbf{H}_{\text{aur}}$, and (2) the convolution of the output signals $\mathbf{l}(t)$ with the estimated acoustic feedback path matrix $\hat{\mathbf{F}}$. The processing latency of the system should be kept as low as possible to ensure a seamless user experience.

B. Real-time convolution

The convolution of the input signals with the synthesis filters $\mathbf{H}_{\text{aur}}$ and the convolution of the output signals $\mathbf{l}(t)$ with the estimated acoustic feedback path matrix $\hat{\mathbf{F}}$ are computationally intensive operations that need to be performed within a predetermined latency budget. For large synthesis filters and many output channels, the convolution process can become infeasible for real-time applications on consumer-grade hardware. To reduce the computational complexity of the convolution process, various methods have been proposed in the literature.

Multichannel block-based time-domain convolution requires a number of operations that is proportional to $\mathcal{O}(n \cdot n_x \cdot n_h)$, where n, n_x and n_h are the number of channels, the block size and the length of the filter, respectively. To accelerate the convolution process, various methods have been proposed in the literature, such as the use of block-based frequency-domain convolution using the Fast Fourier Transform (FFT) [25]. Block-based frequency-domain convolution can be used to reduce the number of operations required for convolution to $\mathcal{O}(n \cdot n_f \log n_f)$, where $n_f \geq n_h + n_x - 1$ is the FFT-size. However, block-based frequency-domain convolution introduces additional processing due to the FFT-size being lower-bounded by $n_h + n_x - 1$ frequency bins. For long filter sizes n_h, this additional processing can introduce latency into the system that could exceed the predetermined latency budget n_x/f_s, which, for real-time auralization systems, is usually only a few milliseconds long.

To further reduce the latency of block-based frequency-domain convolution with long filter lengths n_h, partitioned convolution methods have been proposed in the literature [23], [26], [27]. These methods decompose the filter into smaller sub-filters that can be processed independently using block-based frequency-domain convolution. Partitioned convolution can be implemented in two ways: uniform partitioned convolution and non-uniform partitioned convolution.

In uniform partitioned convolution, the filter of length n_h is divided into K partitions or sub-filters of length n_k, such that $n_h \geq K \cdot n_k$. An input signal with block-length n_x is then convolved with each sub-filter in the frequency-domain using the FFT. Each sub-filter is offset by $k \cdot n_k$ samples, where k denotes the index of the sub-filter. Moreover, when $n_x = n_k$, this delay can be directly realized in the frequency-domain [23]. The resulting output blocks of each sub-filter are then summed using the Overlap-Save (OLS) or Overlap-Add (OLA) method to obtain the output signal in blocks of n_x samples.

Non-uniform partitioned convolution is similar to uniform partitioned convolution, but the filter is divided into K sub-filters of different lengths. This allows for a more efficient use of the available latency budget, as an optimal length can be chosen for each sub-filter to fit within the available latency budget. However, non-uniform partitioned convolution introduces additional complexity due to the varying sub-filter lengths, causing the delay of each sub-filter to be realized in the time domain [23]. In this paper, only the uniform partitioned convolution method is considered, as it is easier

to implement and does not require additional complexity in the implementation.

Partitioned convolution methods are well-suited for parallel processing on multicore processors since the sub-filters can be processed independently of each other. This makes partitioned convolution methods particularly suitable for implementation on GPU architectures. These architectures are designed to perform parallel processing of large amounts of data, at magnitudes higher floating-point performance compared to traditional multicore CPU architectures. In [11], a uniform partitioned convolution method was proposed for real-time auralization systems implemented on a GPU architecture. The implementation was shown to be capable of processing large numbers of sub-filters, while maintaining low latency and high computational efficiency.

Building upon the work of [11], we propose a novel extension by combining acoustic synthesis and feedback cancellation into a unified implementation that employs uniform partitioned convolution on the GPU. This approach is a significant advancement over the original implementation, which only focused on the synthesis stage. By integrating feedback cancellation into the same framework, we can achieve a more efficient and streamlined processing pipeline. In the following section, we provide a detailed description of the uniform partitioned convolution algorithm proposed in [11], which serves as the foundation for our implementation.

IV. Uniform partitioned convolution

The block diagram of the uniform partitioned convolution algorithm used in [11] is shown in Fig. 2. The algorithm is visualized for a single-channel input block $\mathbf{x}_n$ and output block $\mathbf{y}_n$, but can be trivially extended to multiple channels. During initialization, each filter in the matrix $\mathbf{H}_{\text{aur}}$ is split into K sub-filters of length n_k. Typically, the length of each sub-filter is chosen such that $n_k = n_x$, where n_x is the block size of the input signal, as this allows for the delay of each sub-filter to be directly realized in the frequency-domain [11]. As a result, the FFT-size for the input and filters is set to $n_f = 2n_x$, which is the minimum FFT-size when $n_k = n_x$. Each sub-filter is then padded with an additional n_x zeros, before being transformed into the frequency-domain using an n_f-point real-to-complex FFT to obtain $\bar{\mathbf{h}}_k$. The algorithm then comprises three main stages:

1) **Input transformation:** For stream processing, each incoming input block $\mathbf{x}_n$ is copied at the end of an n_f-point sliding window buffer after left-shifting by n_x samples. This is called input packing. This buffer is then transformed into the frequency-domain using the real-to-complex FFT. The resulting frequency spectrum is stored in a buffer of length $\lceil (n_f+1)/2 \rceil$ frequency bins. Since this step does not benefit from parallel processing, it is always performed on the CPU.
2) **Frequency-domain convolution with sub-filters:** The frequency spectrum of the input block $\bar{\mathbf{x}}_n$ is stored into a Frequency-Domain Delay-Line (FDL) buffer containing K blocks. With each new input block, the contents in the FDL are shifted up by one block. Hereafter, the frequency response of each sub-filter is multiplied with the corresponding frequency-domain block in the FDL buffer. Since these operations are independent of each other, they can benefit from parallel processing on the GPU. The resulting spectra are accumulated in an output frequency-domain buffer.
3) **Output transformation:** The accumulated spectrum is transformed back into the time-domain using the complex-to-real Inverse Fast Fourier Transform (IFFT), resulting in n_f output samples. Due to OLS, only the last n_x samples are valid as the output block $\mathbf{y}_n$. This is called output unpacking, and this step is performed on the CPU as it does not benefit from parallel processing.

By implementing the frequency-domain convolution for each sub-filter on a GPU architecture, fast uniform partitioned convolution can be achieved when processing the input signals with the synthesis filters $\mathbf{H}_{\text{aur}}$, and when processing the output signals $\mathbf{l}(t)$ with the estimated acoustic feedback path matrix $\hat{\mathbf{F}}$. This unified approach leverages the parallel processing capabilities of modern GPUs to significantly reduce the computational complexity of the convolution process, especially for long synthesis filters and large numbers of output channels. Moreover, it allows for real-time auralization of complex acoustic environments, such as those found in concert halls and HWS, while maintaining low latency and high computational efficiency.

V. Python implementation

The implementation of the real-time auralization system is done in Python 3.13.1. While Python is generally not regarded as the most efficient language for real-time audio processing, it does allow for rapid prototyping and easy integration with various libraries that can be used to accelerate processing. Such accelerated libraries include `numpy`, `numba`, and `torch`, which offer wrapper functions around optimized C and Compute Unified Device Architecture (CUDA) code to enable real-time performance in Python. The code of this implementation is made available on GitHub[1], a popular open-source code hosting and collaboration platform.

The Python implementation consists of two main components: the uniform partitioned convolution algorithm outlined in Sec. IV and the real-time auralization system that implements both acoustic synthesis and feedback cancellation. The auralization system uses the uniform partitioned convolution implementation to convolve the input signal with a set of time-domain synthesis filters, and the output signal with a set of time-domain feedback cancellation filters. The implementation is designed to be modular and extensible, allowing for easy integration with other components and systems. The code is structured in a way that allows for easy modification and extension, making it suitable for research and development purposes.

Uniform partitioned convolution is implemented in the `PartitionedConvolution` Python class. This class is initialized by a set of C_{out} time-domain filters, a block size n_x, a number of input channels C_{in}, an FFT size n_f, and

[1] https://github.com/hrosseel/GPU-accelerated-auralization

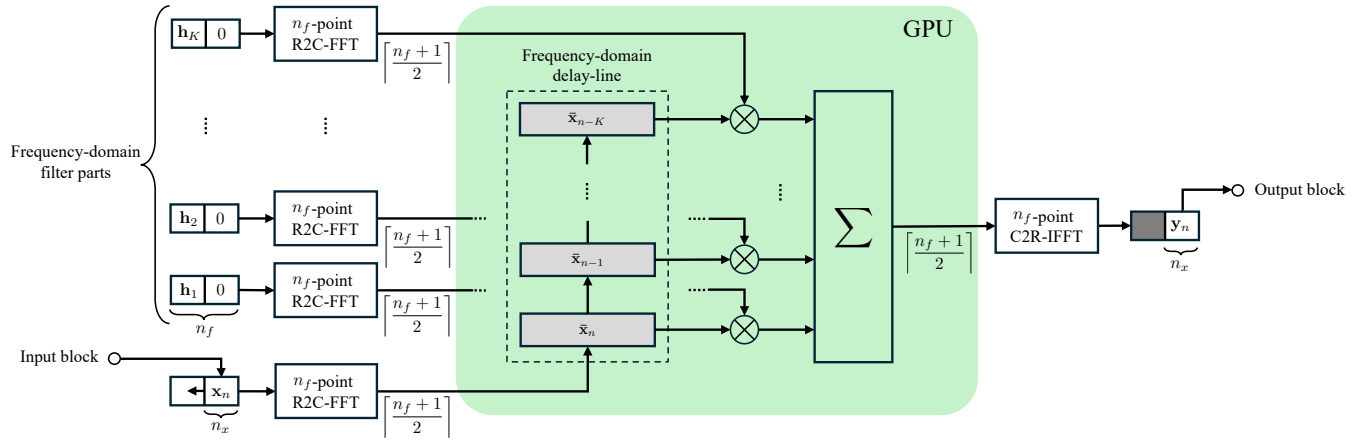

Fig. 2. Block diagram of the uniform partitioned convolution algorithm used in [11]. The algorithm is visualized for a single-channel input block $\mathbf{x}_n$ and output block $\mathbf{y}_n$. The FFT-size for the input and filters was set to $n_f = 2n_x$, where n_x is the block size. The frequency-domain input blocks are denoted by $\bar{\mathbf{x}}_n$. The resulting output block is obtained using the OLS method. The multiplication and summation of the input block spectrum with each of the sub-filter responses can be performed entirely in the frequency-domain on the GPU.

a `device` parameter that specifies whether the processing should be done on the CPU or GPU. The class provides a `convolve` method that inputs a block of samples of size (C_{in}, n_x), and returns a block of filtered output samples of size (C_{out}, n_x). Parameter C_{in} can be set to either 1 or C_{out}. If $C_{\text{in}} = 1$, the single input channel is filtered independently with each of the C_{out} filters, resulting in C_{out} output channels. If $C_{\text{in}} = C_{\text{out}}$, the input signal is filtered with the corresponding filter, resulting in a one-to-one filtering of the input channels with the filters.

The real-time auralization system is implemented in the `PartitionedAuralization` Python class. This class is initialized by a set of C_{out} time-domain synthesis filters, a set of corresponding time-domain feedback cancellation filters, a block size n_x, an FFT size n_f, and a `device` parameter that specifies whether the processing should be done on the CPU or GPU. The class provides an `auralize` method that takes a block of input samples of size $(1, n_x)$ as input, and returns a block of auralized output samples of size (C_{out}, n_x) that can be played back by a loudspeaker system in the reproduction area.

The `PartitionedAuralization` Python class's `auralize` method that internally uses the `convolve` method of the `PartitionedConvolution` class to perform the convolution of the input samples with the synthesis filters, and of the output samples with the feedback cancellation filters. This approach allows for efficient processing as the output of the filtering with the synthesis filters can be directly used as input to the filtering with the feedback cancellation filters, all without the need to move the data from the GPU to the CPU. The estimated feedback path signals are subtracted from the next block of input samples to cancel the contribution of the loudspeaker signals from the input signal.

The current implementation uses a default FFT size of $n_f = 2n_x$. It should be noted that the output of the input block $\mathbf{x}_n$ filtered with the synthesis filters $\mathbf{H}_{\text{aur}}$ could be used directly as frequency-domain input to the filtering operation with the feedback cancellation filters. However, while a computational saving of C_{out} FFT operations could be achieved, this approach does require a change in FFT size from $n_f = 2n_x$ to $n_f = 3n_x$. This increase in FFT-size would result in a computational increase of $(K_{\text{aur}} + K_{\text{fc}})\, C_{\text{out}}\, n_x$ complex multiplications, where K_{aur} and K_{fc} are the number of filter parts for the synthesis and feedback cancellation filters, respectively. This additional cost outweighs the potential saving of C_{out} n_f-point FFT operations when $K_{\text{aur}} + K_{\text{fc}} \geq n_x \log(2n_x)$, which is the case for large synthesis filters.

The performance gain of the GPU implementation is evaluated by comparing the performance to a CPU implementation of the algorithm. The CPU implementation uses the `numba` library to accelerate the convolution operation. The `numba` library is a Just-In-Time (JIT) compiler that translates Python code to optimized machine code at runtime. The CPU implementation is run on multiple CPU threads to further accelerate the processing. Similarly, the GPU implementation of the `PartitionedConvolution` class uses the Python `torch` library to compile and call CUDA code. The CUDA code was written in a separate file and compiled using the `torch` library. A downside of using the CUDA framework instead of alternative frameworks, is that the portability of the implementation is limited to systems with CUDA-compatible GPUs.

A limitation of the current implementation is that only a single input channel is supported for the auralization system. However, this limitation can be overcome by creating multiple instances of the `PartitionedAuralization` class in parallel and summing the respective output channels, thereby enabling support for multiple input channels.

VI. Performance

In this section, we discuss the performance of the implemented partitioned convolution algorithm and real-time auralization system. We compared the performance of the CPU and GPU implementations and analyzed the impact of the block size, number of channels, and filter length on the

performance. The performance was evaluated in terms of the processing time required to convolve a block of input samples with a set of filters.

The workstation used for the performance evaluation was equipped with an AMD Ryzen 9 5950X processor containing 16 CPU cores and 32 threads, 16 GB of RAM, and an NVIDIA RTX A6000 GPU containing 10.752 CUDA cores and 48 GB of GPU memory. The GPU is compatible with the CUDA 11.6 toolkit. The operating system of the workstation was Debian 12 with Linux kernel 5.10.0-26-amd64. In order to stabilize the performance measurements, the system was configured to isolate 12 out of the 16 CPU cores for the real-time auralization system. The CPU isolation was done using the `cgroups` feature of the Linux kernel. The CPU cores were isolated to prevent other processes from interfering with the performance measurements, resulting in a system configuration where the real-time auralization system had exclusive access to 12 CPU cores and the GPU. This led to stable performance measurements.

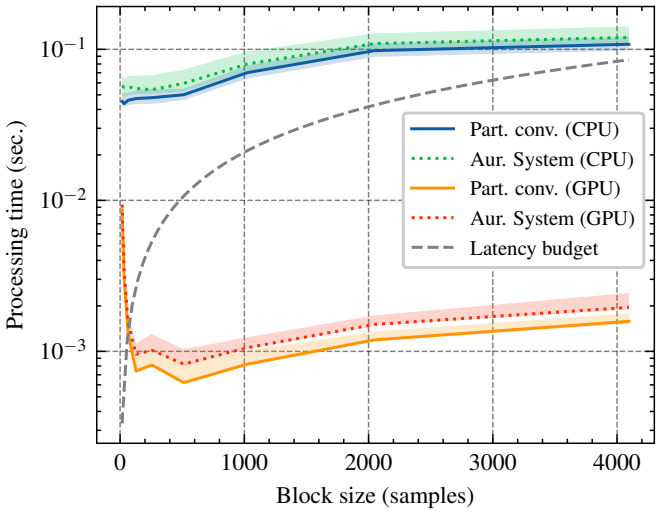

Fig. 3. Performance of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system as a function of the block size.

The processing time of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system were evaluated as a function of the block size n_x, the number of output channels C_{out}, and the filter length. The operating sampling rate of the auralization system was set to 48 kHz, and the default block size was set to $n_x = 128$ samples, which corresponds to a latency budget of 2.6 ms. The default number of output channels was set to $C_{\text{out}} = 32$ channels, and the default filter length was set to 10 seconds, which corresponds with the length of auralization filters that model the acoustics of a highly reverberant church. To evaluate the performance of the system, the synthesis filters, feedback cancellation filters, and input samples were randomly generated from a standard normal distribution. The stochastic nature of the generated filters and input samples avoids potential performance biases that could arise from the use of specific filter coefficients or input samples. The performance was evaluated by measuring the processing time

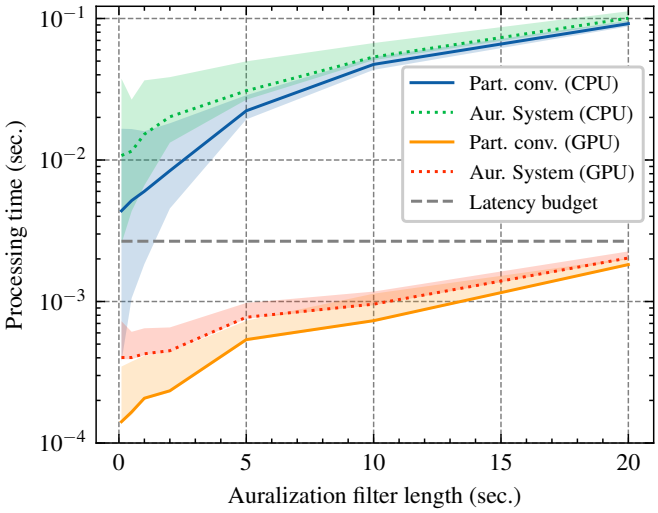

Fig. 4. Performance of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system as a function of the filter length.

required to convolve a single block of input samples with the set of filters, after the system was initialized and the filters had been transformed to the frequency domain and loaded into memory. The processing time was measured using the `time.perf_counter()` function of the Python `time` module, and averaged over 10.000 random trials to obtain a stable performance measurement. In the figures below, the processing time is shown as a function of the block size, auralization filter length, number of output channels, and feedback cancellation filter length. The mean processing time is shown as a solid line for the uniform partitioned convolution algorithm, and as a dashed line for the real-time auralization system implementation. The shaded area around the solid and dashed lines represents the maximum and minimum processing time measured over the 10.000 random trials.

In Fig. 3, the performance of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system is shown as a function of the block size, which varied from 16 to 4096 samples corresponding to a latency budget of 0.3 ms to 85.3 ms, respectively. To achieve real-time performance, the processing time of the real-time auralization system should be less than the latency budget. It can be seen that for uniform partitioned convolution and auralization on the CPU, real-time performance is not achieved for any of the block sizes. For the GPU implementation of uniform partitioned convolution and auralization, real-time performance is achieved for block sizes higher than 128 samples. For these block sizes, the mean performance of the GPU implementation is up to 60 higher than the CPU implementation.

In Fig. 4, the performance is shown as a function of the filter length, which is varied from 0.1 to 20 seconds. It can be seen that the processing time of both the CPU and GPU implementations increases as the filter length increases. With the CPU implementation of the real-time auralization system, real-time filtering is not achieved, as the processing time

exceeds the latency budget for all filter lengths. However, the performance of the GPU implementation is able to achieve real-time performance for filter lengths up to 20 seconds.

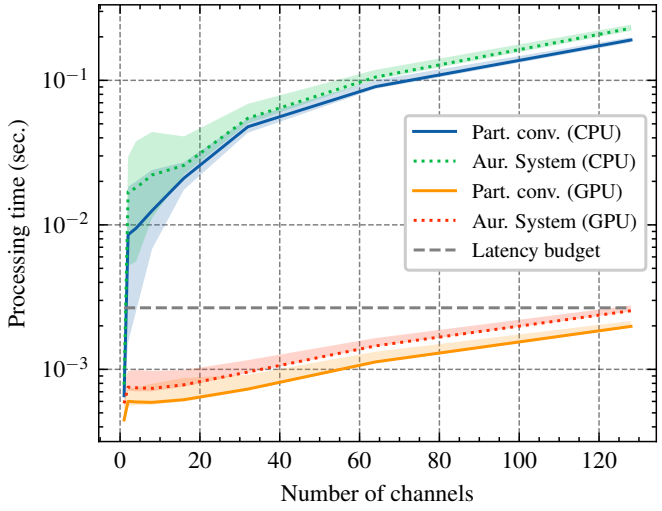

Fig. 5. Performance of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system as a function of the number of output channels.

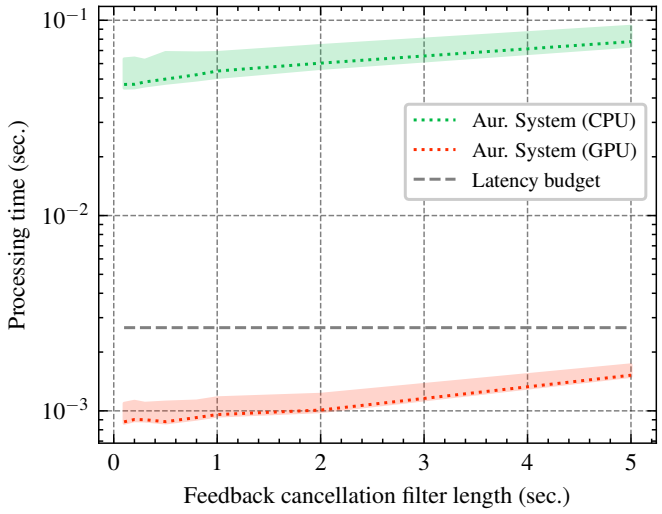

Fig. 6. Performance of the CPU and GPU implementations of the partitioned convolution algorithm and the real-time auralization system as a function of the feedback cancellation filter length.

The performance as a function of the number of output channels C_{out} is shown in Fig. 5. The performance is evaluated for a number of output channels that is varied from 1 to 128 channels. It can be seen that the processing time of both the CPU and GPU implementations increases as the number of channels increases. However, the performance of the GPU implementation decreases at a slower rate than the CPU implementation. Real-time filtering is achieved for up to a single output channel on the CPU and up to 120 output channels on the GPU.

Finally, the performance is shown as a function of the feedback cancellation filter length in Fig. 6, which is varied from 0.1 to 5 seconds. It can be seen that the overall performance of the real-time auralization system is not significantly impacted by the feedback cancellation filter length of up to 5 seconds. This is due to the fact that the feedback cancellation filters are typically much shorter than the synthesis filters.

VII. Conclusion

In this paper, we presented a real-time implementation of a GPU-accelerated loudspeaker-based auralization system for the interactive synthesis of large room acoustics with integrated feedback cancellation. The system was designed to be flexible and applicable to a wide range of loudspeaker-based auralization scenarios, with virtually no limitation on the length of the synthesis filters and the number of channels in the reproduction setup. The implementation was evaluated in terms of processing time by comparing the performance of the GPU and CPU implementations. It was shown that the GPU implementation achieves a performance gain of up to 60 times compared to the CPU implementation, enabling real-time auralization at 48 kHz for synthesis filters with a duration of up to 20 seconds and 32 output channels. The results demonstrated that the proposed implementation was capable of performing real-time auralization with minimal latency, making it suitable for interactive applications in large room acoustics. While the Python implementation of the system was not optimized for real-time performance, it was shown to achieve real-time performance using accelerated libraries. Future work will focus on further optimizing the system for real-time performance, and on interactive perceptual evaluation of the system.

VIII. Acknowledgments

This research work was carried out at the ESAT Laboratory of KU Leuven, in the frame of KU Leuven internal funds C3/23/056 “HELIXON: Hybrid, efficient, and liquid interpolation of sound in extended reality” and C14/21/075 ”A holistic approach to the design of integrated and distributed digital signal processing algorithms for audio and speech communication devices”, FWO SBO Project The sound of music - Innovative research and valorization of plainchant through digital technology” (S005319N), FWO Large-scale research infrastructure ”The Library of Voices - Unlocking the Alamire Foundation’s Music Heritage Resources Collection through Visual and Sound Technology” (I013218N), and SBO Project ”New Perspectives on Medieval and Renaissance Courtly Song” (S005525N). The research leading to these results has received funding from the Flemish Government under the AI Research Program and from the European Research Council under the European Union’s Horizon 2020 research and innovation program / ERC Consolidator Grant: SONORA (no. 773268). This paper reflects only the authors’ views and the Union is not liable for any use that may be made of the contained information.

References

[1] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, “Fifty Years of Artificial Reverberation,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421–1448 (2012).

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "More Than 50 Years of Artificial Reverberation," presented at the *Proc. AES 60th Int. Conf. Dereverberation and Reverberation of Audio, Music, and Speech*, pp. 1–12 (2016).

[3] M. Vorländer, *Auralization* (Springer Berlin, Heidelberg, 2020).

[4] K. Schiltz, "Church and Chamber: The Influence of Acoustics on Musical Composition and Performance," *Early Music*, vol. 31, no. 1, pp. 65–78 (2003).

[5] K. Ueno and H. Tachibana, "Experimental study on the evaluation of stage acoustics by musicians using a 6-channel sound simulation system," *Acoust. Sci. Technol.*, vol. 24, pp. 130–138 (2003).

[6] K. Ueno, K. Kato, and K. Kawai, "Effect of Room Acoustics on Musicians' Performance. Part I: Experimental Investigation with a Conceptual Model," *Acta Acust United Ac*, vol. 96, pp. 505–515 (2010).

[7] A. Bassuet, "Acoustics of early music spaces from the 11th to 18th century: Rediscovery of the acoustical excellence of medium-sized rooms and new perspectives for modern concert hall design," *J. Acoust. Soc. Am.*, vol. 115, no. 5, pp. 2582–2582 (2004).

[8] A. Lindau, "The Perception of System Latency in Dynamic Binaural Synthesis," presented at the *Proc. Int. Conf. Acoustics (NAG/DAGA)*, pp. 1063–1066 (2009).

[9] T. van Waterschoot and M. Moonen, "Fifty Years of Acoustic Feedback Control: State of the Art and Future Challenges," *Proc. of the IEEE*, vol. 99, no. 2, pp. 288–327 (2011).

[10] J. S. Abel, E. F. Callery, and E. K. Canfield-Dafilou, "A Feedback Canceling Reverberator," presented at the *Proc. 21st Int. Conf. Digital Audio Effects (DAFx '18)*, pp. 100–106 (2018).

[11] F. Wefers and J. Berg, "High-Performance Real-Time FIR-Filtering using Fast Convolution on Graphics Hardware," presented at the *Proc. 13th Int. Conf. Digital Audio Effects, (DAFx '10)*, pp. 1–8 (2010).

[12] L. Savioja, D. Manocha, and M. Lin, "Use of GPUs in room acoustic modeling and auralization," presented at the *Proc. Int. Symposium Room Acoustics (ISRA 2010)*, pp. 1–7 (2010).

[13] B. Cowan and B. Kapralos, "Spatial sound for video games and virtual environments utilizing real-time GPU-based convolution," presented at the *Proc. 2008 Conference on Future Play: Research, Play, Share*, pp. 166–172 (2008).

[14] B. Cowan and B. Kapralos, "GPU-Based One-Dimensional Convolution for Real-Time Spatial Sound Generation," *Loading...: The Journal of the Canadian Game Studies Association*, vol. 3, no. 5, pp. 1–14 (2009).

[15] N. Tsingos, "Using Programmable Graphics Hardware for Auralization," presented at the *Proc. EAA Symposium on Auralization*, pp. 1–10 (2009).

[16] D. A. Mauro, *On Binaural Spatialization and the use of GPGPU for Audio Processing*, Ph.D. thesis, Università degli Studi di Milano (2011).

[17] J. A. Belloch, M. Ferrer, A. Gonzalez, F. Martinez-Zaldivar, and A. M. Vidal, "Headphone-based Spatial Sound with a GPU Accelerator," *Procedia Computer Science*, vol. 9, pp. 116–125 (2012).

[18] 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).

[19] J. A. Belloch, A. Gonzalez, F. J. Martínez-Zaldívar, and A. M. Vidal, "Real-time massive convolution for audio applications on GPU," *The Journal of Supercomputing*, vol. 58, no. 3, pp. 449–457 (2011).

[20] J. A. Belloch, M. Ferrer, A. Gonzalez, J. Lorente, and A. M. Vidal, "GPU-based WFS systems with mobile virtual sound sources and room compensation," presented at the *Proc. AES 52nd Int. Conf. Sound Field Control*, pp. 1–8 (2013).

[21] R. Ranjan and W.-S. Gan, "Fast and efficient real-time GPU based implementation of wave field synthesis," presented at the *Proc. 2014 IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '14)*, pp. 7550–7554 (2014).

[22] J. A. Belloch, A. Gonzalez, E. S. Quintana-Ortí, M. Ferrer, and V. Välimäki, "GPU-Based Dynamic Wave Field Synthesis Using Fractional Delay Filters and Room Compensation," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 2, pp. 435–447 (2017).

[23] F. Wefers, *Partitioned convolution algorithms for real-time auralization*, Ph.D. thesis, RWTH Aachen University, Institute of Technical Acoustics, Aachen, Germany (2014).

[24] H. Rosseel and T. van Waterschoot, "A state-of-the-art review on acoustic preservation of historical worship spaces through auralization," *Signal Processing*, vol. 234, pp. 1–25 (2025).

[25] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301 (1965).

[26] T. G. Stockham, "High-speed convolution and correlation," presented at the *Proc. of the April 26-28, 1966, Spring Joint Computer Conference*, pp. 229–233 (1966).

[27] A. Torger and A. Farina, "Real-Time Partitioned Convolution for Ambiophonics Surround Sound," presented at the *Proc. 2001 IEEE Workshop Appls. Signal Process. Audio Acoust. (WASPAA '01)*, pp. 195–198 (2001).