Software Design for Interactive Room Acoustic Simulation

Samuel Clapp, Manuel G. Hornung, Sebastian Pods, and Bernhard U. Seeber
Audio Information Processing, Technical University of Munich, 80290 Munich, samuel.clapp@tum.de

Introduction

In the Audio Information Processing Group at the Technical University of Munich, the Simulated Open Field Environment (SOFE) is employed for simulating, rendering, and presenting room acoustical environments to listeners for psychoacoustic research [1]. Room acoustics are simulated using the Image Source Method and rendered for playback to a horizontal ring of 96 loudspeakers. Each image source is rendered to the nearest loudspeaker, and image sources outside the azimuthal plane can be rendered either to the loudspeaker of the same azimuth, or to the one which lies on the same cone of confusion as the elevated source.

The next generation of SOFE, known as the Real-Time Simulated Open Field Environment (rtSOFE), is currently being developed. The hardware is installed in a newly constructed anechoic chamber measuring 4 x 6 x 10 meters, with an array of 60 loudspeakers: 36 in the azimuthal plane, as well as rings of 12 below and 12 above the azimuthal plane. In addition, new software has been developed allowing for the presentation of dynamic acoustic scenes with which listeners can interact.

System Goals

The principal use of the system is for conducting psychoacoustic research with normal-hearing and hearing-impaired listeners, as well as cochlear implant users. The goal is to bring more "life-like" listening scenarios into the laboratory. Auditory perception in these scenarios can then be studied in a controlled manner, which is difficult to accomplish in field studies.

Psychoacoustic tests are often relatively passive exercises for test subjects, where they are presented with stimuli, and then asked for a response. However, this is not how we normally navigate the real world. We move through our environment, interact with it, and attempt to control those aspects which we are able. Thus, we desire a system that can incorporate changes to the acoustic environment in real time with low latency, and offer an accurate and convincing simulation of the movement of both sound sources and receivers.

Listeners also sometimes act, themselves, as the sound source. The most common example is speaking, either as a monologue (i.e. a teacher in classroom) or in dialogue (i.e. conversing with others in a restaurant). The sound can also be musical, either singing or with an instrument. Finally, some listeners (especially those who are blind) employ a strategy of echolocation by emitting short clicks and listening to the signal returned by the environment to glean information about the positions of walls and objects. So, the system should also be able to handle real-time sound input from test subjects with low latency.

Technical Requirements

The most important requirement for the system is a high degree of physical accuracy. The simulated environments should accurately represent the acoustics that would be found in a real environment of the same design. The system needs to be both real-time (sound samples are processed and output at the same rate at which they come in) and low-latency (as short a delay as possible between when changes to the simulation are detected, and when the output reflects those changes). The real-time and low-latency requirements are important for establishing a sense of realism, and for ensuring that participants behave normally in the simulated scenarios, without artificially slowing down their movements if they sense that the system cannot keep up with rapid changes. Satisfying all of these requirements is important so that results obtained in laboratory experiments have validity in "real world" scenarios.

The system should be able to simulate not just rectangular rooms, but rooms of arbitrary geometries, including slanted walls, protruding elements, and objects inside the room. It should also be able to handle impulse responses of up to 3 seconds (typical for some larger spaces like concert halls and churches) at 44.1 kHz on 60 independent channels.

This system thus requires two main components. The first is a fast method for generating highly accurate room impulse responses for simulated rooms, which can respond quickly to changes in input parameters. The second is a real-time convolution engine with low input/output latency and the ability to transition smoothly between different impulse responses to offer a convincing and realistic presentation of source and receiver movement.

System Architecture

Based on the development of previous generations of SOFE, a modular architecture has been employed for the organization of the system as a whole. Each component of the system can be run on its own dedicated PC, including one for the room simulation, and one for the real-time convolution and output. The system components can then communicate using standard networking protocols (UDP and TCP).

Room Acoustics Simulation

Computational Method

The Image Source Method (ISM) is used, where the position of the sound source is repeatedly mirrored over the room boundaries to determine the temporal and spatial information of the room reflections [2]. The system uses ISM up to high orders (often up to 50-200 depending on the size of the room and reflectivity of its surfaces), with the lower orders representing early reflections and the higher orders

representing the late reverberation (via the high temporal density and wide spatial distribution of higher-order image sources). A temporal jitter of 5-10% is applied to higher-order image sources to avoid the spectral coloration resulting from a highly regularized geometrical pattern of image sources [1].

For a rectangular room, one can calculate the positions of image sources up to an arbitrarily high order via spatially repeating patterns [2]. However, for non-rectangular rooms, a visibility test is required to determine if each reflection reaches the receiver [3].

In the ISM, first-order image sources (i.e. those representing a single wall reflection from source to receiver) are calculated by reflecting the original sound source location over all of the room boundaries. Second-order image sources are then calculated by reflecting the first-order image sources over the room boundaries, and so on. Thus, each image source spawns a new set of image sources of one order higher. This process can be thought of as generating "trees" of image sources that grow exponentially with increasing order.

Such a simulation could be allowed to run forever, and thus a stopping criterion is required. The following stopping criteria can be applied [1]:

- **Maximum order**: how many times an image source can be reflected over the room boundaries
- **Attenuation**: as image sources lose energy with each successive reflection, as well as increasing distance from the receiver, one can define a threshold in dB below which sources will no longer be calculated
- **Maximum number of sources**: the total number of image sources calculated
- **Maximum distance**: the maximum distance an image source can be located from the receiver; divided by the speed of sound, this yields the maximum time for the length of the impulse response
- **Number of invisible parents**: invisible image sources can still spawn visible ones of higher orders. However, a branch with many invisible sources is often unlikely to yield more visible sources further down the line, so employing this as a stopping criterion can save memory and computation time.

At a minimum, one of these criteria must be set by the user to run the simulation. However, multiple stopping criteria can be specified, with the most conservative one taking precedence during run-time.

Computational Efficiency

Several methods were employed to increase the computational efficiency of the simulation, to reach the fast computation times desired for our real-time applications. *OpenMP*, an open-source toolbox for parallelization, was used to distribute the computational load over multiple threads [4]. The ISM computations are well-suited to parallelization, since the calculation of each new image source is only based on the parent source. Depending on which stopping criteria are specified, different branches of the image source "tree" may end at different orders. Therefore, the computational workload is redistributed amongst the threads after each order has been calculated.

One performance bottleneck for ISM simulations for arbitrary room geometries is the visibility test. The traditional way to compute this is to calculate a cross-product, which requires six multiplications per vertex of the room surface in question. However, in this implementation, the PNPOLY algorithm was used, which only requires one multiplication and one division per vertex, saving computation time on this step [5, 6].

Finally, the simulation can be run in "merger" mode, where two different instances of the simulation with different parameters are run simultaneously, with the calculated impulse responses merged at the output. This allows for running the simulation with the early part of the impulse response being re-calculated at a higher update rate (where small changes in source and receiver positions are more audible) and the later part of the impulse response at a lower rate, for instance, to save on computational load, but still deliver a perceptually valid simulation. This modularity will also allow the system to incorporate other methods in the future, such as wave-based or stochastic simulation methods.

Performance Evaluation

The performance of the room simulation software was evaluated using a geometric model of a seminar room at the Technical University of Munich [7]. The room model contained 24 surfaces. Performance was evaluated with calculations of different image source orders on an Intel Core i7 desktop PC with different numbers of threads. In all cases, the room simulation was constantly receiving updates, so whenever one simulation finished, a new one began immediately.

Results are shown in Table 1, and given in number of sources calculated per millisecond. Orders 3, 5, and 8 were examined, resulting in 1k, 80k, and 63M sources, respectively.

For the simplest calculations (order 3), a single thread actually performs the best, while the benefits of more threads are realized at higher orders. When the total number of image sources is in the tens of the thousands, the four- and eight-thread simulations exhibit similar performance. The eight-thread simulation shows a benefit compared to the others when simulating a very high number of image sources, in the millions.

Table 1: Desktop PC, "coarse" model

Threads	ISM order (Number of sources)		
	3 (1k)	5 (80k)	8 (63M)
1	8912	18400	15750
4	8267	34480	27720
8	6933	35040	36540

Real-Time Convolution

Computational Method

The second important component of this system is a real-time convolution engine. One possible implementation for real-time convolution is the direct-form implementation of a finite impulse response (FIR) filter. The benefit of this approach is low input/output latency, as it can be computed on a sample-by-sample basis. The disadvantage is that it is not very efficient for long impulse responses. One of the goals of this system was that it could be used for impulse responses up to 3 seconds, and at a sampling frequency of 44.1 kHz, which would yield a filter of over 130,000 taps.

Convolution can also be computed in the frequency domain, using point-by-point multiplication. This is a much more efficient computational process, especially for long impulse responses. However, the main drawback is that a block of samples is required before the processing can take place, leader to higher input/output latency.

The method employed here is based on one first proposed by Gardner [8]. The first part of the impulse response is convolved using direct-form filtering. This allows the system to accumulate enough input samples to then apply frequency-domain convolution for the later part of the impulse response, while maintaining a low input/output latency. The frequency-domain convolution is achieved using blocks of samples, with the first blocks after the direct-form portion containing half of the samples used in the direct-form filter. The size of the blocks used for the frequency-domain convolution can increase for later sections of the impulse response, leveraging the efficiency gains of larger block sizes while maintaining a low input/output latency.

One other important requirement for our system is that it can switch smoothly between different room impulse responses. To accomplish this, the program must convolve the relevant input samples with two different impulse responses simultaneously – the “old” one and the “new” one. Then, the output signals are cross-faded to achieve a smooth transition. Thus, the system requires sufficient processing overhead from the “default” status with a single impulse response, as it may be called upon to process twice as many samples any time a new impulse response is received.

Computational Efficiency

The calculations of the forward and inverse Fourier Transforms needed for executing the frequency-domain convolution are accomplished using the FFTW library, which has been shown to have excellent performance in this task[9]. The complex multiplication (used to convolve the signals in the frequency domain) is accomplished using the Intel Advanced Vector Extensions (AVX), available on the most recent Intel processors, which also significantly reduces computation time for this step [10]. Parallelization was achieved by assigning the different portions of the impulse response (direct-form filter and the three FFT block sizes) to different threads running on different processor cores.

Performance Evaluation

System performance was evaluated on an Intel Core i7 desktop PC [11]. The convolver was given 64-channel impulse responses of 3 seconds in length at a sampling frequency of 44.1 kHz. The length of the callback function was set at 32 samples, and a new impulse response was given to the convolver at every callback.

The partitioning scheme used a 512-tap filter for the direct-form convolution, and frequency domain block sizes of 256, 1024, and 4096 samples (with 6, 6, and 31 blocks, respectively).

The results are shown in Table 2. The performance of each portion of the impulse response was analyzed separately, with average and maximum processing times given for each section. Processing times using this partitioning scheme were found to be well within the requirements. Thus, this real-time convolution engine meets the goal of simultaneous convolution of 60 channels with 3-second, dynamically updated impulse responses and low input/output latency.

Table 2: Convolution performance

	Filter Portion			
	FIR	FFT 1	FFT 2	FFT 3
I/O Samples	32	128	512	2048
Avg. Exec. Time	330 µs	2,5 ms	10 ms	40 ms
Max. Exec. Time	420 µs	5 ms	13 ms	55 ms
Available Time	726 µs	6 ms	23 ms	93 ms

Full System

The full system, of which a schematic diagram is shown in Figure 1, can be controlled via a single script written in Matlab, Python, or any other programming language/framework that can send UDP messages. At startup, this script can send initialization commands (via UDP) to the room simulation software to set the room geometry, simulation parameters, and angular positions of the rendering loudspeakers. Initialization commands sent to the real-time convolution software include the maximum length of the impulse response, the partitioning scheme, the length of the callback, number of loudspeaker channels, and the length of the crossover between impulse responses.

These software components also receive UDP messages from the control script at run-time. The room simulation software can receive new source and receiver positions, while the convolution engine receives control commands for the audio stream, such as starting, stopping, and changing the source of the audio.

The final communication component is the impulse responses, one per loudspeaker channel, that are sent from the room simulation to the convolution software. This communication occurs via TCP, which includes an extra “handshake” to ensure the accuracy of the transmitted data.

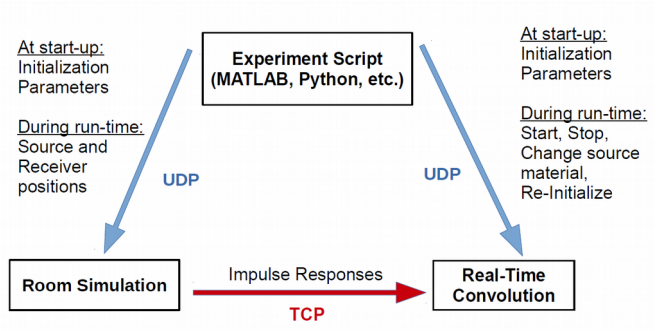

Figure 1: Full system schematic diagram

Conclusion

This paper described the design of software for rtSOFE, the newest iteration of the Simulated Open-Field Environment [1]. This system can simulate virtual, interactive acoustic environments in real-time at low latency. These environments can be employed for psychoacoustic research with normal-hearing and hearing-impaired listeners and cochlear implant users.

Acknowledgements

This work was supported by the Bernstein Center for Computational Neuroscience Munich, BMBF 01 GQ 1004B. Manuel G. Hornung contributed the design, coding, and performance evaluation of the room simulation software for his Master's Thesis, and Sebastian Pods contributed the same for the real-time convolution software for his Bachelor's Thesis, in the Audio Information Processing group at the Technical University of Munich.

References

[1] B. U. Seeber, S. Kerber, and E. R. Hafter, "A System to Simulate and Reproduce Audio-Visual Environments for Spatial Hearing Research," *Hear. Res.*, vol. 260, no. 1–2, pp. 1–10, 2010.

[2] J. A. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943-950, 1979.

[3] J. Borish, "Extension of the image model to arbitrary polyhedra," *J. Acoust. Soc. Am.*, vol. 75, no. 6, pp. 1827-1836, 1984.

[4] OpenMP application program interface 4.0, URL: `http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf`

[5] PNPOLY – point inclusion in polygon test, URL: `http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html`

[6] A Quijada Gomariz, "Acoustics Simulation using Graphics Hardware based on the Image Source Model," B.Sc. Thesis, T.U. Munich, 2014.

[7] M. G. Hornung, "Implementation and Optimization of a Software Framework for interactive Room Acoustics Simulation," M.Sc. Thesis, T.U. Munich, 2015.

[8] W. G. Gardner, "Efficient Convolution without Input-Output Delay," *J. Audio Eng. Soc.*, vol. 43, no. 3, pp. 127-136, 1995.

[9] FFTW Home Page, URL: `http://www.fftw.org`

[10] Intel Intrinsics Guide, URL: `http://software.intel.com/sites/landingpage/IntrinsicsGuide/#`

[11] S. Pods, "Echtzeit Faltung von mehrkanaligen Impulsantworten," B.Sc. Thesis, T.U. Munich, 2016.