EVERTims: Open source framework for real-time auralization in architectural acoustics and virtual reality

David Poirier-Quinot, Brian F.G. Katz, Markus Noisternig

EVERTIMS: OPEN SOURCE FRAMEWORK FOR REAL-TIME AURALIZATION IN ARCHITECTURAL ACOUSTICS AND VIRTUAL REALITY

David Poirier-Quinot
Acoustic and Cognitive Spaces Group
IRCAM, CNRS, Sorbonne University, UPMC
Paris 6, UMR 9912, STMS
Paris, France

Brian F.G. Katz
Institut Jean Le Rond d'Alembert
Sorbonne Universités
UPMC Univ Paris 06, CNRS
Paris, France

Markus Noisternig
Acoustic and Cognitive Spaces Group
IRCAM, CNRS, Sorbonne University, UPMC
Paris 6, UMR 9912, STMS
Paris, France

ABSTRACT

This paper presents recent developments of the EVERTims project, an auralization framework for virtual acoustics and real-time room acoustic simulation. The EVERTims framework relies on three independent components: a scene graph editor, a room acoustic modeler, and a spatial audio renderer for auralization. The framework was first published and detailed in [1, 2]. Recent developments presented here concern the complete re-design of the scene graph editor unit, and the `C++` implementation of a new spatial renderer based on the JUCE framework. EVERTims now functions as a Blender add-on to support real-time auralization of any 3D room model, both for its creation in Blender and its exploration in the Blender Game Engine. The EVERTims framework is published as open source software: `http://evertims.ircam.fr`.

1. INTRODUCTION

Auralization is "*the process of rendering audible, by physical or mathematical modeling, the sound field of a source in space [...] at a given position in the modeled space*" [3]. Room acoustic auralization generally relies on Room Impulse Responses (RIR), either measured or synthesized. An RIR represents the spatio-temporal distribution of reflections in a given room upon propagation from an emitter (source) to a receiver (listener). Figure 1 details several conceptual components of an RIR.

Auralization has some history of being used in architectural design, *e.g.* for the subjective evaluation of an acoustic space during the early stage of its conception [4]. It also serves for perceptually evaluating the impact of potential renovation designs on existing spaces. Auralization is often used as a supplement to objective parameter metrics [5, 6, 7]. While previously limited to industry and laboratory settings, auralization is now receiving particular attention due to the emergence of Virtual Reality (VR) technologies [8, 9]. Lately, most research have been driven by the need for 1) real-time optimization and 2) accuracy and realism of the simulated acoustics [10, 11, 12], where these two goals drive developments in opposing directions. The phenomenon is similar to the race for real-time lighting engines and the advent of programmable shaders [13].

An overview of the many available techniques for simulating the acoustics of a given geometry has previously been presented in [4, 14, 15], detailing both ray-based [16, 17] and wave-based techniques [18, 19]. Ray-based techniques are based on a high frequency approximation of acoustic propagation as geometric rays, ignoring diffraction and modal effect. They usually outperform wave-based techniques regarding calculation speed.

A further distinction is made here between ray-based techniques, referring to the family of methods based on the geometrical acoustics hypothesis, and ray-tracing techniques [20], one of its subfamily. As detailed in [2], the EVERTims room acoustic modeler relies on an "image source" technique [17] (ray-based). The implementation is based on a beam tracing algorithm [2], focused on real-time estimation of specular early reflections.

A number of geometrical acoustic simulation programs have been developed having auralization capabilities, either tailored for architectural acoustics [21, 22] or acoustic game design [23, 24] (3DCeption Spatial Workstation[1], Audioborn[2], DearVR[3], VisiSonics RealSpace 3D[4]). The level of realistic detail achieved is typically inversely proportional to the real-time capabilities. The ambition in developing EVERTims has been to combine the accuracy of the former and the performance of the later. It does not claim to outperform any of these softwares, but rather to provide a reliable, simple, and extensible tool for acousticians, researchers, and sound designers.

The main contribution of this paper is to present the evolution of the EVERTims project since its first publication in [2]. Section 2 gives an overview of the architecture of the EVERTims real-time auralization framework. Section 3 introduces the Blender add-on, which replaces the previously implemented VirChor scene graph editor. Only minor revisions have been made to the room acoustic modeler that is briefly described in Section 4. The implementation of the new auralization and spatial audio rendering unit is described in Section 5. Section 6 summarizes the current state of the framework and outlines future developments.

[1] Two Big Ears website: `www.twobigears.com/spatworks`
[2] Audioborn website: `www.audioborn.com`
[3] DearVR website: `www.dearvr.com`
[4] VisiSonics website: `www.visisonics.com`

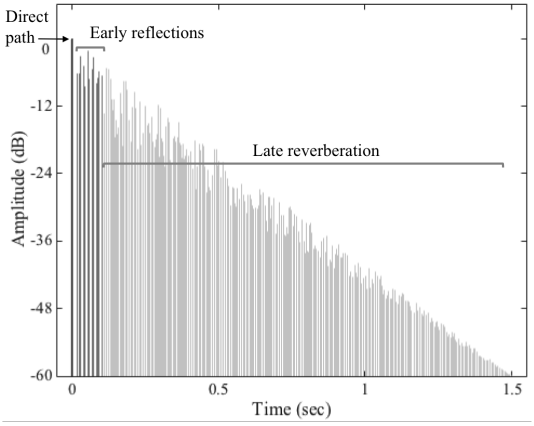

Figure 1: *Theoretical plot illustrating the conceptual components of a RIR: direct path, early reflections, and late reverberation.*

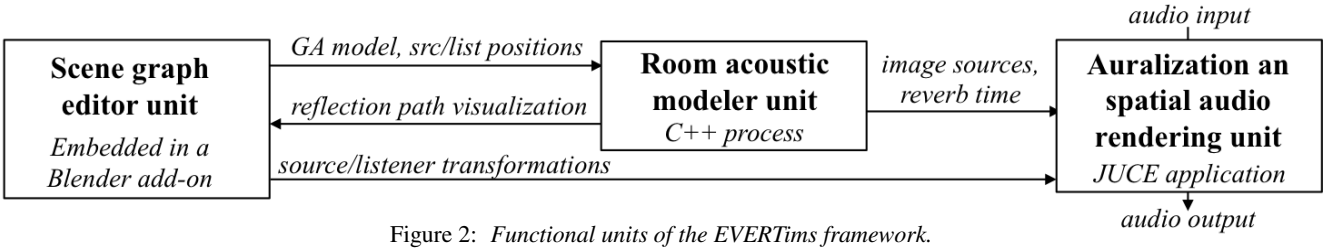

Figure 2: *Functional units of the EVERTims framework.*

2. FRAMEWORK OVERVIEW

Figure 2 shows the functional units of the EVERTims framework. The scene graph editor handles the room geometry and the acoustic properties of the wall surface materials, referred to as the room's Geometrical Acoustic (GA) model, as well as the user interaction. This editor, implemented as a Blender add-on, provides a Graphical User Interface (GUI) from which the room acoustic modeler, the auralization unit, and the real-time auralization process (*i.e.* initiating the data exchange between the functional units) can be started. The room acoustic modeler runs as a background subprocess, the auralization unit as a standalone application.

In order to support the interchangeability of algorithms the main functional units are fully encapsulated and self-contained. The communication and data exchange is based on the Open Sound Control (OSC) protocol [25]. This allows the distribution of sub-tasks to different computers, or to run the entire environment on a single computer. During the auralization, the Blender add-on streams GA model information, along with listener and source positions, to the acoustic modeler. Based on these data, the acoustic modeler builds a list of image sources, corresponding to a minimum reflection order defined at start-up (see Section 4). These image sources, along with an estimated reverberation time for the current GA model, are then sent to the auralization unit. The Blender add-on also streams source and listener transformation matrices to the auralization unit. Based on the data issued from both the add-on and the modeler, the auralization unit generates and spatialises image sources as early reflections, as well as constructs the late reverberation. An additional callback can be initiated by the add-on to display the results of the acoustic modeler simulation as reflection paths drawn in the 3D scene for monitoring.

EVERTims uses an iterative refinement procedure and computes higher reflection orders progressively. Whenever there is a change in the geometry or a sound source position, an approximate beam-tree up to the minimum reflection order is computed. The visible paths are then sent to the auralization unit. When there is no change, it continues to compute the solution up to the next reflection order until the chosen maximum order is reached. When a listener moves, visibility tests are computed for all the paths in the beam-trees for that listener and changes are sent to the auralization unit. Changes in source/listener orientation do not affect the beam-tree and visibility of paths; there is no need to perform any recalculation. A change to the geometry or the source position is computationally expensive, as it requires a new computation of the underlying beam tree, reconstructing the beam-tree up to the minimum order, and beginning the iterative order refinement of the solution once again.

For architects and acousticians, the core feature of the framework is to provide both a reactive auralization during room design and exploration, providing an interactive rendering based on the actual geometry of the space, and an accurate auralization for final room acoustics assessment. The same framework allows for integration into game engines for real-time applications. At any time, intermediate rendering results can be exported as an RIR in different audio formats (*e.g.* binaural, Ambisonic), suitable for use in any convolution based audio renderer.

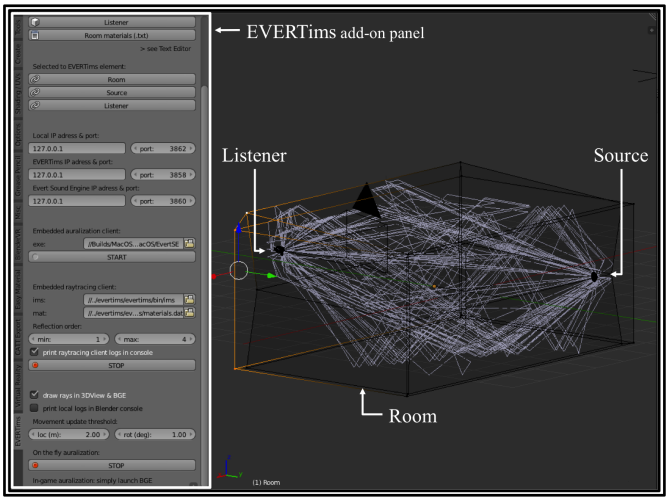

Figure 3: *Blender interface during an EVERTims auralization session. The EVERTims add-on is displayed on the left panel. The "reflection path" debug mode has been activated to monitor the output of the acoustic modeler (reflection paths) for the current geometry.*

3. SCENE GRAPH EDITOR AND 3D VISUALIZATION

The EVERTims scene graph editor and 3D visualizations are handled in Blender. The Blender add-on is written in Python, and integrated in the 3D View Toolbox. The add-on itself handles EVERTims specific data (GA, assets import, OSC setup, etc.) and processes.

To start the auralization, the add-on requires four EVERTims-specific elements: *room*, *source*, *listener*, and *logic*. These elements can either be defined from existing objects in the scene, or imported from an assets library packaged with the add-on.

The "source" and "listener" can be attached to any object in the 3D scene. Also the "room" can be defined from any mesh geometry; the add-on does not apply any checks on the room geometry (*e.g.* convexity, closed form, etc.), allowing EVERTims to work for open or sparse scenes as well as closed spaces. An acoustic material can be assigned to each of the room mesh faces. The wall surface materials are defined in the EVERTims materials library, imported from the assets pack, and are shared with the room acoustic modeler unit (see Section 4). Each acoustic material is defined by a set of absorption coefficients (10 octave bands from 32 Hz to 16 kHz) and a frequency independent diffusion coefficient. The materials library consists of a human-readable text file which is easy to modify and extend.

The "logic" object handles the parameters of the Blender add-on when EVERTims is running in *game engine mode* for real-time auralization. This mode uses the Blender Game Engine for building virtual walkthroughs. Its counterpart, the EVERTims *edit mode*, auralizes the room model in real-time during design and creation phases. Both modes rely on the same `evertims` Python module, called either from add-on GUI callbacks or logic bricks attached to the EVERTims logic object. Figure 3 illustrates the Blender interface during an EVERTims auralization session in edit mode.

During an auralization session, callbacks of the `evertims` Python module stream GA related information along with listener and source positions to the acoustic modeler. In edit mode, any room geometry modifications and changes of the wall surface materials are forwarded to the acoustic modeler for real-time updates of the auralization unit. Spatial and temporal thresholds can be defined to throttle the update mechanism, *e.g.*, the minimum amount of movement required for either source or listener to send an update status in order to control communication traffic for example in the case of head-tracker jitter. For monitoring and debugging, the simulation results can be visualized as rays in the 3D scene. This is depicted in Figure 3. In addition, a low-priority thread can be started to output local or acoustic modeler subprocess logs to the Blender console.

4. ROOM ACOUSTIC MODELER

This section gives a brief description of the room acoustic modeler unit, which is further detailed in [2]. The modeler unit is a console application, implemented in `C++`. Upon reception of the room geometry and listener & source positions, the modeler unit constructs a beam tree for the current scene geometry. The maximum height of this tree is defined by the highest simulated reflection order passed as a parameter to the modeler. From this tree, a list of image sources is generated and sent to the auralization unit. For visualization, a list of acoustic paths is sent back to the Blender add-on (shown in Figure 3).

The iterative order refinement of the solution (see above) ensures a reactive auralization for dynamic scenes, and high accuracy for the auralization of scenes with fixed source(s) position(s) and room geometry.

Each specular reflection path is characterized by its direction of arrival relative to the listener, propagation delay, absorption due to frequency-dependent material properties, and air absorption. Results of the room acoustic model consisting of visible reflection paths and their accumulated material attenuation are sent to the audio renderer. The reflection path message also contains the "first reflection point"

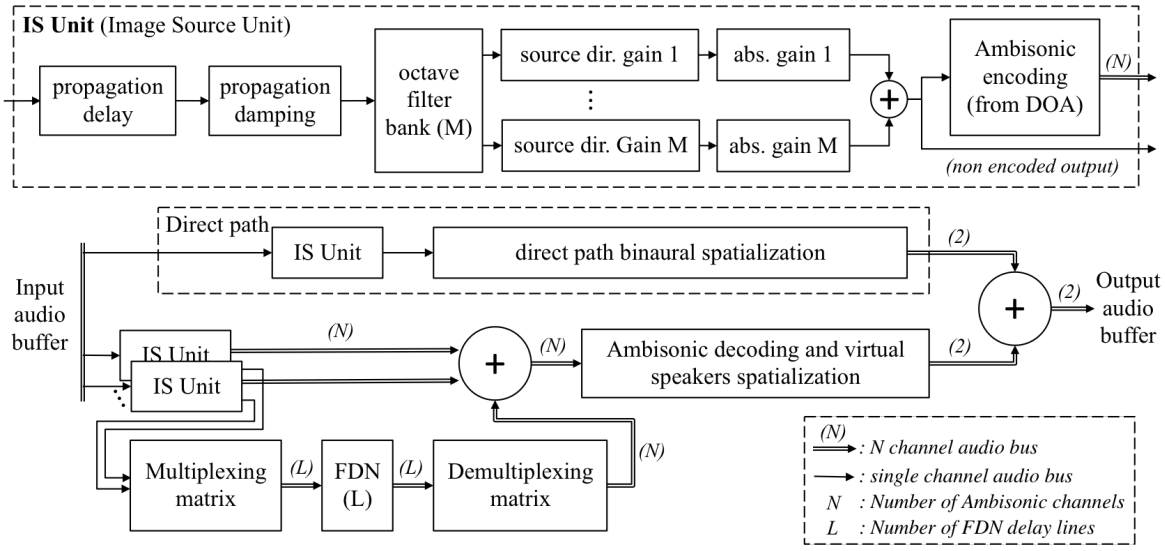

Figure 4: *General architecture of the auralization engine.*

for implementing the source directivity (see Section 5). The generic form of an image source message is:

$$\text{pathID order } r1_x\ r1_y\ r1_z\ rN_x\ rN_y\ rN_z \text{ dist } abs_0 \ldots abs_9 \text{ diff} \quad (1)$$

where $[r1_x, r1_y, r1_z]$ and $[rN_x, rN_y, rN_z]$ are the position vectors of the first and last reflections respectively, $dist$ is the length of the whole reflection path, abs_M is the absorption coefficient for the M^{th} octave band, and *diff* is the frequency independent diffusion coefficient (which is currently not used by the audio renderer).

As the reflection paths can only be computed up to a limited order in real-time, a statistical model, currently based on Sabine's formula [26], is used to approximate the late reverberation time of the current room. The reverberation time is estimated per octave band and sent to the auralization unit's feedback delay network processor.

The room acoustic modeler runs on multiple threads: one for input processing, one for updating visibility changes, and one for calculating new solutions. The modeler supports multi-source & mutli-listener scenarios. It can achieve interactive update rates for a moving listener while calculating up to seventh order reflections, with a model containing less than one thousand polygons. A complete characterisation of the EVERTims modeler unit, including performance assessment and an evaluation of its simulation fidelity as compared to other auralization engines, is presented in [2].

5. SPATIAL AUDIO RENDERING FOR AURALIZATION

The general architecture of the auralization unit is detailed in Figure 4. The unit is implemented in `C++` using the JUCE framework[5] and packaged as a standalone application. At present, the auralization unit is designed for binaural playback over headphones and processes either an audio file or a microphone input signal.

To each image source sent by the acoustic modeler is associated a delayed tap of the current audio input (*e.g.* audio file buffer). To this tap is applied an attenuation proportional to the length of the whole reflection path of the image source. A frequency specific gain is then applied, computed based on the abs_M coefficients in (1). The number of bands of the filter-bank used for signal frequency decomposition can be defined between 3 and 10. The 3-band option is designed for simulations with a large number of image sources and/or architectures with reduced CPU resources. To further reduce CPU consumption, the filter-bank implementation is based on successively applied low-pass filters (see *e.g.* the implementation described in [27]) rather than a series of parallel bandpass filters.

The $[r1_x, r1_y, r1_z]$ position vector of (1) along with the source orientation are used to compute each image source specific Direction of Departure (DOD). Based on this DOD and a directivity diagram loaded from a SOFA *GeneralTF* file, an octave-band specific directivity gain is applied to the image source audio buffer. The current implementation proposes a basic set of predefined directivity diagrams (omnidirectional, cardiod, etc.).

The resulting audio buffers of each image source are encoded in third order Ambisonics and summed to create a single Ambisonics stream sound-field. The sound-field is then decoded to binaural, based on the virtual speaker approach [28]. For line of sight scenarios, the audio tap of the direct path is handled to a dedicated binaural encoder rather than to the Ambisonic processing unit. The objective of this direct encoding, already applied in the previous version of the auralization unit [2], is to further increase the accuracy of the perceived sound source location. Both decoding schemes are based on filters dynamically loaded from a *SimpleFreeFieldHRIR* SOFA file. The interpolation of incomplete HRIR measurement grids is not yet supported by the auralization unit, being handled via Matlab routines at the moment.

[5]JUCE website: `www.juce.com`

The auralization unit uses a Feedback Delay Network (FDN) to simulate the late reverberation part of the RIR [29]. While the acoustic modeler efficiency allows for dense RIR simulation for static geometries, sparser image source reflection information RIRs are to be expected for dynamic geometries, forcing the modeler into a "low reflection order" mode. The FDN implementation is based on a matrix of 16 parallel feedback delay lines. The audio buffers resulting from the computation of the early reflections described above are fed to this network based on a random distribution. FDN input, output, and feedback matrices are tailored to limit inter-channels correlation and optimise network density [30]. The FDN decay time is frequency specific. Underlying FDN delays and gains are thus defined based on the room response time, estimated by the room acoustic modeler (see [30] for a detailed discussion). FDN delays are mutually prime while ensuring a sufficiently high mode density in all frequency bands to avoid any "ringing tone" effect. FDN outputs are encoded into first order Ambisonics and added to the main Ambisonics stream sound-field (see above) for spatial audio playback. Higher-order Ambisonics encoding for high-resolution binaural playback is planned for future updates of the standalone auralization unit.

A log display is available in the auralization application to monitor all currently registered source, listener, image sources, and room reverberation times. An additional option allows to export the current RIR to either stereo (binaural) or Ambisonic formats in an audio file.

6. CONCLUSION

This paper presented the latest developments of the EVERTims framework. The aim of the underlying project is to design an accurate auralization tool to support research and room acoustic design. Besides accuracy, the framework is focused on real-time rendering for the auralization of dynamic environments. Its integration as a Blender add-on allows for real-time assessment of room acoustics during its creation.

Each unit of the EVERTims framework is available as an open source project. These units can be used separately for integration in any other framework. Sources, tutorials, recordings and exchange protocol information are available on the EVERTims website: `http://evertims.ircam.fr`.

These latest developments mainly concerned the re-design of the EVERTims framework interface and the integration of its different components. The novel Blender add-on simplifies the control of the overall auralization simulation, providing end-users with a state of the art mesh editing tool in the process. The implementation of the auralization and spatial audio rendering unit as a JUCE `C++` graphical application, rather than a PureData patch, should simplify future maintenance and cross-platform compatibility. The new auralization unit now supports the SOFA format [27] to import either HRIR or source directivity patterns.

Foreseen short-term developments include multi-source and multi-listener integration along with cross-platform support. The acoustic modeler unit already supports multi-source & multi-listener, the feature only lacks its counterpart in the auralization unit. The auralization engine for its part is already available as a cross-platform application, thanks to the versatility of the JUCE framework. Only minor modifications to the Blender add-on will be required to support both features. A graphical editor to handle acoustic material creation and modification will be added to the add-on, along with an editor for source directivity.

7. REFERENCES

[1] Markus Noisternig, Lauri Savioja, and Brian F.G. Katz, "Real-time auralization system based on beam-tracing and mixed-order ambisonics," *The Journal of the Acoustical Society of America*, vol. 123, no. 5, pp. 3935–3935, 2008.

[2] Markus Noisternig, Brian F.G. Katz, Samuel Siltanen, and Lauri Savioja, "Framework for real-time auralization in architectural acoustics," *Acta Acustica United with Acustica*, vol. 94, no. 6, pp. 1000–1015, 2008.

[3] Mendel Kleiner, Bengt-Inge Dalenbäck, and Peter Svensson, "Auralization-an overview," *J. Audio Eng. Soc*, vol. 41, no. 11, pp. 861–875, 1993.

[4] Peter Svensson and Ulf R Kristiansen, "Computational modelling and simulation of acoutic spaces," in *22nd Audio Engineering Society Conference*. Audio Engineering Society, 2002.

[5] John S Bradley, "Review of objective room acoustics measures and future needs," *Applied Acoustics*, vol. 72, no. 10, pp. 713–720, 2011.

[6] Brian F.G. Katz and Eckhard Kahle, "Design of the new opera house of the Suzhou Science & Arts Cultural Center," in *Western Pacific Acoustics Conf. (WESPAC)*, 2006, pp. 1–8.

[7] Brian F.G. Katz and Eckhard Kahle, "Auditorium of the Morgan library, computer aided design and post-construction results," in *Intl. Conf. on Auditorium Acoustics*, Oslo, Oct. 2008, Institute of Acoustics, vol. 30, pp. 123–130.

[8] Barteld N.J. Postma, David Poirier-Quinot, Julie Meyer, and Brian F.G. Katz, "Virtual reality performance auralization in a calibrated model of Notre-Dame Cathedral," in *Euroregio*, Porto, June 2016, pp. 6:1–10.

[9] Brian F.G. Katz, David Poirier-Quinot, and Jean-Marc Lyzwa, "Interactive production of the "virtual concert in Notre-Dame"," in *19th Forum Intl. du Son Multicanal (FISM)*, Paris, Nov. 2016.

[10] Michael Vorländer, *Auralization: fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*, Springer Science & Business Media, 2007.

[11] Lauri Savioja, "Real-time 3d finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *13th Int. Conf on Digital Audio Effects*, 2010, vol. 1, p. 75.

[12] Dirk Schröder, *Physically based real-time auralization of interactive virtual environments*, vol. 11, Logos Verlag Berlin GmbH, 2011.

[13] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman, *Real-time rendering*, CRC Press, 2008.

[14] Michael Vorländer, *Auralization*, Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality. Springer, 2008.

[15] Lauri Savioja and U Peter Svensson, "Overview of geometrical room acoustic modeling techniques," *The Journal of the Acoustical Society of America*, vol. 138, no. 2, pp. 708–730, 2015.

[16] Asbjørn Krokstad, Staffan Strom, and Svein Sørsdal, "Calculating the acoustical room response by the use of a ray tracing technique," *Journal of Sound and Vibration*, vol. 8, no. 1, pp. 118–125, 1968.

[17] Jont B Allen and David A Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[18] Brian Hamilton, *Finite Difference and Finite Volume Methods for Wave-based Modelling of Room Acoustics*, Ph.D. thesis, University of Edinburgh, 2016.

[19] Stefan Bilbao and Brian Hamilton, "Wave-based room acoustics simulation: Explicit/implicit finite volume modeling of viscothermal losses and frequency-dependent boundaries," *Journal of the Audio Engineering Society*, vol. 65, no. 1/2, pp. 78–89, 2017.

[20] K Heinrich Kuttruff, "Auralization of impulse responses modeled on the basis of ray-tracing results," *Journal of the Audio Engineering Society*, vol. 41, no. 11, pp. 876–880, 1993.

[21] Bengt-Inge L Dalenbäck, "Room acoustic prediction based on a unified treatment of diffuse and specular reflection," *The journal of the Acoustical Society of America*, vol. 100, no. 2, pp. 899–909, 1996.

[22] Graham M Naylor, "Odeon, another hybrid room acoustical model," *Applied Acoustics*, vol. 38, no. 2-4, pp. 131–143, 1993.

[23] Regis Faria and Joao Zuffo, "An auralization engine adapting a 3d image source acoustic model to an ambisonics coder for immersive virtual reality," in *28th Audio Engineering Society Conference*. Audio Engineering Society, 2006, pp. 1–4.

[24] Wolfgang Ahnert and Rainer Feistel, "Ears auralization software," in *Audio Engineering Society Convention*. Audio Engineering Society, 1992, pp. 894–904.

[25] Matthew Wright, Adrian Freed, et al., "Open soundcontrol: A new protocol for communicating with sound synthesizers.," in *ICMC*, 1997, pp. 1–4.

[26] Wallace Clement Sabine and M David Egan, "Collected papers on acoustics," *The Journal of the Acoustical Society of America*, vol. 95, no. 6, pp. 3679–3680, 1994.

[27] Yukio Iwaya, Kanji Watanabe, Piotr Majdak, Markus Noisternig, Yoiti Suzuki, Shuichi Sakamoto, and Shouichi Takane, "Spatially oriented format for acoustics (SOFA) for exchange data of head-related transfer functions," in *Meeting of the Institute of Electronics, Information, and Communication Engineers (IECE) of Japan*, 2014, pp. 19–23.

[28] Markus Noisternig, Thomas Musil, Alois Sontacchi, and Robert Holdrich, "3d binaural sound reproduction using a virtual ambisonic approach," in *International Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems*. IEEE, 2003, pp. 174–178.

[29] Manfred R. Schroeder, "Natural sounding artificial reverberation," *J. Audio Eng. Soc*, vol. 10, no. 3, pp. 219–223, 1962.

[30] Jean-Marc Jot and Antoine Chaigne, "Digital delay networks for designing artificial reverberators," in *Audio Engineering Society Convention*, 1991, vol. 39, p. 383.