Wayverb: A Graphical Tool for Hybrid Modelling Auralisation

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Master of Arts

Matthew Reuben Thomas

January 2017

This is the abstract.

Contents

Acknowledgements

This research was supported by the Creative Coding Lab at the University of Huddersfield.

I would like to thank my supervisor, Alexander Harker (PhD; Lecturer), for his invaluable input and patience.

Introduction

The aim of impulse response synthesis is to simulate the reverberant properties of a space without having to physically build anything. This is useful for a variety of applications: architects need to be able to evaluate the acoustics of a building before construction begins; sound editors for film sometimes need to mix in recordings which were not made on location; electronic musicians like to conjure imaginary or impossible spaces in their music, and virtual-reality experiences must use audio cues to convince the user that they have been transported to a new environment.

Unfortunately, software allowing the synthesis of accurate binaural impulse responses is not currently widely available. Often, software produced for research purposes is not made public. Such software that *is* available generally suffers from one or more of an array of issues.

Most software relies only on fast geometric methods, which are inaccurate, especially at low frequencies. Conversely, programs opting to use more accurate wave-modelling methods require long time periods, in the order of days, or significant computing power to run.

Licensing is also an problem. Most room-acoustics packages are the product of years of combined research by multiple contributors, which is only made viable by releasing the software commercially. However, this inhibits further research, as the code is not freely available. This model also limits users to those able to pay, restricting widespread adoption.

When software is made available freely, often the user experience suffers. Code requires manual compilation, or perhaps can only be run from a textual interface, or else the project is outdated and unmaintained.

The Wayverb project provides a solution to these problems, by making available a graphical tool for impulse response synthesis. It combines several simulation techniques, providing an adjustable balance between speed and accuracy. It is also free to download, can be run immediately on commodity hardware, and the source code can be used and extended under the terms of the GNU General Public License (GPL).

This thesis will begin by examining common methods of room simulation and the software which implements these methods, explaining why particular techniques were chosen for Wayverb. Then, each of the chosen techniques will be explored in depth, along with a description of their implementation. The procedure for producing a single impulse response from the outputs of multiple modelling techniques will be detailed. Two extensions to the basic room acoustics model will be described, namely frequency-dependent reflections at boundaries, and microphone/head-related transfer function (HRTF) simulation. The project will be evaluated, and finally, avenues for future development will be examined.

1 Context

Overview

Room acoustics algorithms fall into two main categories: *geometric*, and *wave-based* [1]. Wave-based methods aim to solve the wave equation numerically, simulating the actual behaviour of sound waves within an enclosure. Geometric methods instead make some simplifying assumptions about the behaviour of sound waves, which result in faster but less accurate simulations. These assumptions generally ignore all wave properties of sound, choosing to model sound as independent *rays*, *particles*, or *phonons*.

The modelling of waves as particles has found great success in the field of computer graphics, where *ray-tracing* is used to simulate the reflections of light in a scene. The technique works well for simulating light because of the relatively high frequencies of the modelled waves. The wavelengths of these waves - the wavelengths of the visible spectrum - will generally be many times smaller than any surface in the scene being rendered, so wave phenomena have little or no visible effect.

The assumption that rays and waves are interchangeable falls down somewhat when modelling sound. The wavelengths of sound in air range from 17m to 0.017m for the frequency range 20Hz to 20KHz, so while the simulation may be accurate at high frequencies, at low frequencies the wavelength is of the same order as the wall surfaces in the scene. Failure to take wave effects such as interference and diffraction into account at these frequencies therefore results in noticeable approximation error [2].

In many cases, some inaccuracy is an acceptable (or even necessary) trade-off. Wave-modelling is so computationally expensive that using it to simulate a large scene over a broad spectrum could take weeks on consumer hardware. This leaves geometric methods as the only viable alternative. Though wave-modelling been studied for some time [3], and even applied to small simulations of strings and membranes in consumer devices such as keyboards, it is only recently, as computers have become more powerful, that these techniques have been seriously considered for room acoustics simulation.

Given that wave-based methods are accurate, but become more expensive at higher frequencies, and that geometric methods are inexpensive, but become less accurate at lower frequencies, it is natural to combine the two models in a way that takes advantage of the desirable characteristics of each. That is, by using wave-modelling for low-frequency content, and geometric methods for high-frequency content, simulations may be produced which are accurate across the entire spectrum, without incurring massive computational costs.

Characteristics of Simulation Methods

A short review of simulation methods will be given here. For a detailed survey of methods used in room acoustics, see [4].

The following figure (1.1) shows the relationships between the most common simulation methods. The advantages and disadvantages of each method will be discussed throughout the remainder of this section.

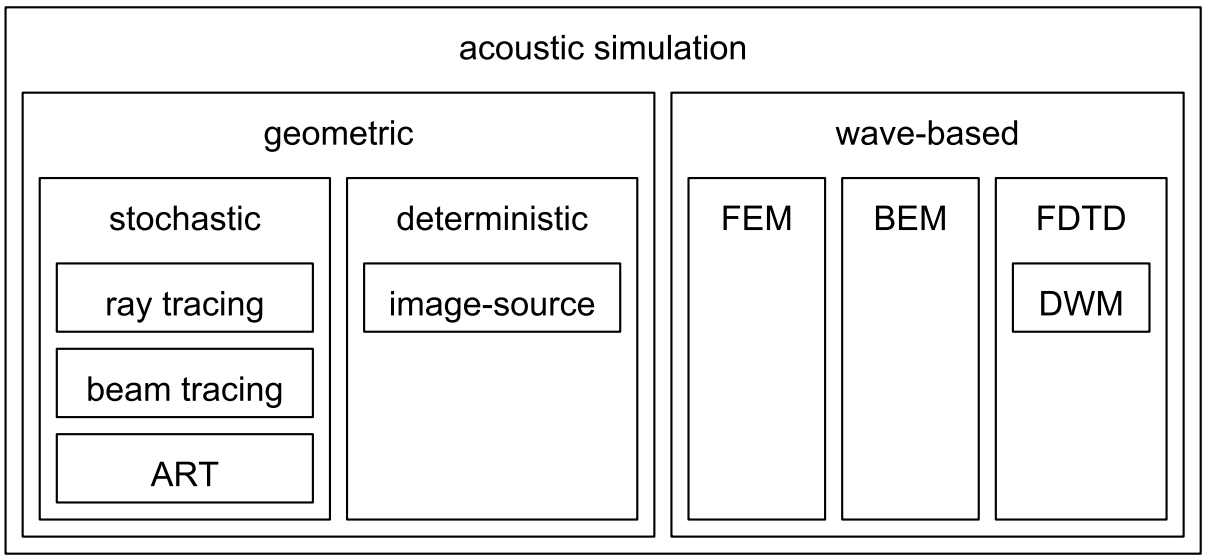

Figure 1.1: An overview of different acoustic simulation methods, grouped by category.

Geometric

Geometric methods can largely be grouped into two categories: *stochastic* and *deterministic*.

Stochastic methods are generally based on statistical approximation via some kind of Monte Carlo algorithm. Such algorithms are approximate by nature. They aim to randomly and repeatedly sample the problem space , combining the results from multiple trials so that they converge upon the correct answer. The balance of quality and speed can be adjusted in a straightforwward manner, simply by adjusting the number of samples taken.

In room acoustics, stochastic algorithms may be based directly on reflection paths, using *ray tracing* or *beam tracing*, in which rays or beams are considered to transport acoustic energy around the scene. Alternatively, they may use a surface-based technique, such as *acoustic radiance transfer* (ART), in which surfaces are used as intermediate stores of acoustic energy.

Surface-based methods, especially, are suited to real-time simulations (i.e. interactive, where the listener position can change), as the calculation occurs in several passes, only the last of which involves the receiver object. This means that early passes can be computed and cached, and only the final pass must be recomputed if the receiver position changes.

The main deterministic method is the *image source* method, which is designed to calculate the exact reflection paths between a source and a receiver. For shoebox-shaped rooms, and perfectly rigid surfaces, it is able to produce an exact solution to the wave equation. However, by its nature, it can only model specular (perfect) reflections, ignoring diffuse and diffracted components. For this reason, it is inexact for arbitrary enclosures, and unsuitable for calculating reverb tails, which are predominantly diffuse. The technique also becomes very expensive beyond low orders of reflection. The naive implementation reflects the sound source against all surfaces in the scene, resulting in a set of *image sources*. Then, each of these image sources is itself reflected against all surfaces. For high orders of reflection, the required number of calculations quickly becomes impractical. For these reasons, the image source method is only suitable for early reflections, and is generally combined with a stochastic method to find the late part of an impulse response.

For a detailed reference on geometric acoustic methods, see [2].

Wave-based

The main advantage of wave-based methods is that they inherently account for wave effects like diffraction and interference [5], while geometric methods do not. This means that these wave-based methods are capable of accurately simulating the low-frequency component of a room impulse-response, where constructive and destructive wave interference form *room modes*. Room modes have the effect of amplifying and attenuating specific frequencies in the room impulse response, and produce much of the subjective sonic "colour" or "character" of a room. Reproducing these room modes is therefore vital for evaluating the acoustics of rooms such as concert halls and recording studios, or when producing musically pleasing reverbs.

Wave-based methods may be derived from the *Finite Element Method* (FEM), *Boundary Element Method* (BEM) or *Finite-Difference Time-Domain* (FDTD) method. The FEM and BEM may be known together as *element methods*.

The FEM is an iterative numerical method for finding natural resonances of a bounded enclosure. It models the air pressure inside the enclosure using a grid of interconnected nodes, each of which represents a mechanical system with a single degree of freedom. The interconnectedness of the nodes leads to a set of simultaneous equations, which can be solved for displacement at each node, and then the solved equations can be used to calculate pressure values at certain elements. The BEM is similar, but models nodes on the surface of the enclosure, instead of within it. This in turn allows it to model unbounded spaces. [6]

The FDTD method works by dividing the space to be modelled into a regular grid, and computing changes in some quantity at each grid point over time. The formula used to update each grid point, along with the topology of the grid, may be varied depending on the accuracy, efficiency, and complexity required by the application. FDTD methods are generally applied to problems in electromagnetics, but a subclass of the FDTD method known as the *Digital Waveguide Mesh* (DWM) is often used for solving acoustics problems.

The FDTD process shares some characteristics with the element methods. They all become rapidly more computationally expensive as the maximum output frequency increases [7]. They also share the problem of discretisation or quantisation, in which details of the modelled room can only be resolved to the same accuracy as the spatial sampling period. If a large inter-element spacing is used, details of the room shape will be lost, whereas a small spacing will greatly increase the computational load.

The major advantage of FDTD over element methods is that it is run directly in the time domain, rather than producing frequency-domain results, which in turn affords a much simpler implementation.

The main disadvantage of the FDTD method is that it is susceptible to *numerical dispersion*, in which wave components travel at different speeds depending on their frequency and direction, especially at high frequencies. Several techniques exist to reduce this error, such as oversampling the mesh [8], using different mesh topologies [9], [10], and post-processing the simulation output [11]. Oversampling further increases the computational load of the simulation, while using different topologies and post-processing both introduce additional complexity.

Despite its drawbacks, the FDTD method is generally preferred for room acoustics simulation [7], probably due to its straightforward implementation, intuitive behaviour, and its ability to directly produce time-domain impulse responses.

TODO note that FDTD is embarrassingly parallel

Existing Software

Searching online and in the literature uncovers a handful of programs for acoustic simulation. The following table (1.1) shows a selection which is not exhaustive, but which is felt to be representative.

Table 1.1: Some of the most prominent tools for acoustic simulation.

Name	Type	Availability
Odeon [12]	Geometric	Commercial
CATT-Acoustic [13]	Geometric	Commercial
Olive Tree Lab [14]	Geometric	Commercial
EASE [15]	Geometric	Commercial
Auratorium [16]	Geometric	Commercial
RAVEN [17]	Geometric	None
RoomWeaver [18]	Waveguide	None
EAR [19]	Geometric	Free
PachydermAcoustic [20]	Geometric	Free
Parallel FDTD [21]	Waveguide	Free
i-Simpa [22]	Geometric, extensible	Free

All commercial acoustics programs found use geometric techniques, probably because they are fast to run, and can often be implemented to run interactively, in real-time. However, low-frequency performance is a known issue with these programs. For example, the FAQ page for the Odeon software [23] notes that:

> For Odeon simulations as with real measurements, the source and receiver should be at least 1/4th wave length from the walls. But at the very lowest resonance of the room the level can change a lot from position to position without Odeon being able to predict it. For investigation of low frequency behavior (resonances), indeed Odeon is not the tool.

Clearly there is a need for wave-modelling acoustics software, which can accurately predict low frequency behaviour. However, such software seems to be somewhat rarer than geometric acoustics software. Of the two wave-modelling programs listed, only one is generally available, which must additionally be run from Python or Matlab scripts. This is a good approach for research software, but would probably not be straightforward for users with limited programming experience.

At time of writing, December 2016, it appears that no generally-available (commercially or otherwise) piece of software has taken the approach of combining wave-modelling and geometric methods, although this technique is well-known in the literature [1], [24]–[28].

Research Aims

With all this in mind, it appears that there is a requirement for a program which combines geometric and wave-based methods to produce simulations which are accurate across the audible spectrum. Rather than focussing on performance, or interactive simulation (which is already implemented in the commercial software above), such a program should strive towards accuracy first, and performance second. To be useful to end-users, the program should have a graphical interface, though a scripting or library interface might be provided for research purposes. Finally, the program would ideally be free and open-source, to maximise adoption and to aid future research and collaboration. The goal of the Wayverb project was to produce a program which satisfied these requirements.

Strategy

Chosen Simulation Techniques

As the image-source method is well-suited to finding early reflections, and stochastic methods are reasonably accurate at computing the more diffuse late reflections, it made sense to combine these two methods for high-frequency simulation. Specifically, a simple ray tracing method was chosen over a phonon- or surface-based method for the late-reflection simulation, for two reasons. Firstly, ray tracing is broadly discussed in the literature [29]–[33], so would not require a great deal of experimentation to implement. Secondly, ray tracing has the property of being an *embarrassingly parallel* algorithm, because each individual ray can be simulated entirely independently, without requiring communication or synchronisation. By running the algorithm on graphics hardware, which is designed to run great numbers of calculations in parallel, all rays could be simulated in one go, yielding much greater performance than processing each ray sequentially.

A logistical reason for choosing the image-source and ray tracing solution for high-frequency modelling was that the author had previously implemented such a system for an undergraduate project. It was hoped that much of the code from that project could be re-used, but it transpired that much of the code was unsuitable or incorrect (!), so that the majority was completely re-written. The author was, however, able to re-use much of the knowledge and experience gained from the previous project, which would not have been possible if a completely new stochastic method had been introduced.

For low-frequency simulation, a FDTD-based DWM model was chosen. There is a great deal of writing on this method [6], [8], [34]–[36], it is relatively simple to implement, and shares with ray tracing the characteristic of being embarrassingly parallel. Each element in the waveguide mesh can be updated individually and simultaneously, which it was hoped would yield performance benefits.

An in-depth description of the algorithms implemented is given in the Image-Source, Ray Tracer, and Waveguide sections. The following figure (1.2) shows how the outputs from the different methods work together to produce a broadband impulse response.

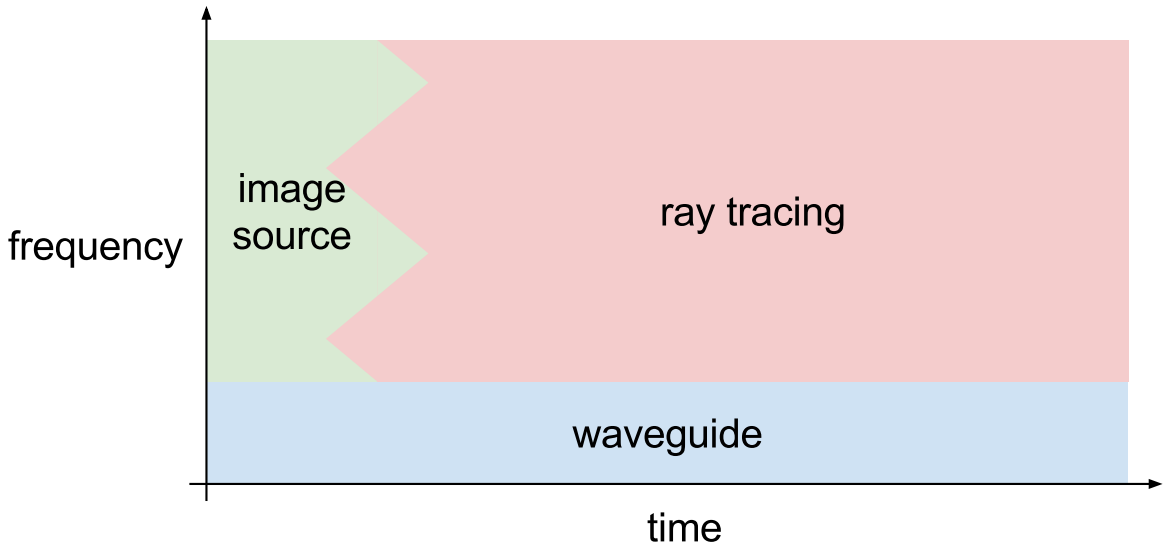

Figure 1.2: The structure of a simulated impulse response.

Deciding on the simulation techniques led to three questions:

- To produce a final output, the three simulations must be automatically mixed in some way. How can this be done?
- Binaural simulation requires some method for direction- and frequency-dependent attenuation at the receiver. How can receivers with polar patterns other than omnidirectional be modelled

consistently in all three simulation methods?

- The reverb time and character depends heavily on the nature of the reflective surfaces in the scene. How can frequency-dependent reflective boundaries be modelled consistently in all methods?

These questions will be discussed in the Hybrid, Microphone Modelling, and Boundary Modelling sections respectively.

Chosen Technology

The programming language chosen was C++. For acceptable performance in numerical computing, a low-level language is required, and for rapid prototyping, high-level abstractions are necessary. C++ delivers on both of these requirements, for the most part, although its fundamentally unsafe memory model does introduce a class of bugs which do not really exist in languages with garbage collection, borrow checking, or some other safety mechanism.

OpenCL was chosen for implementing the most parallel parts of the simulation. The OpenCL framework allows a single source file to be written, in a C-like language, which can target either standard *central processing units* (CPUs), or highly parallel *graphics processing units* (GPUs). The main alternative to OpenCL is CUDA, which additionally can compile C++ code, but which can only target Nvidia hardware. OpenCL was chosen as it would allow the final program to be run on a wider variety of systems, with fewer limitations on their graphics hardware.

The only deployment target was macOS. This was mainly to ease development, as maintaining software across multiple platforms is often time-consuming. macOS also tends to have support for newer C++ language features than Windows. Visual Studio 2015 for Windows still does not support all of the C++11 language features [37], while the Clang compiler used by macOS has supported newer C++14 features since version 3.4 [38], released in May 2014 [39]. Targeting a single platform avoids the need to use only the lowest common denominator of language features. As far as possible, the languages and libraries have been selected to be portable if the decision to support other platforms is made in the future. Once Windows fully supports C++14, it should be possible to port the program with a minimum of effort.

The following additional libraries were used to speed development. They are all open-source and freely available.

GLM Provides vector and matrix primitives and operations, primarily designed for use in 3D graphics software, but useful for any program that will deal with 3D space.

Assimp Used for loading and saving 3D model files in a wide array of formats, with a consistent interface for querying loaded files.

FFTW3 Provides Fast Fourier Transform routines. Used mainly for filtering and convolution.

Libsndfile Used for loading and saving audio files, specifically for saving simulation results.

Libsamplerate Provides high-quality sample-rate-conversion routines. Waveguide simulations are often run at a relatively low sample-rate, which must then be adjusted.

Gtest A unit-testing framework, used to validate small individual parts of the program, and ensure that changes to one module do not cause breakage elsewhere.

Cereal Serializes data to and from files. Used for saving program configuration options.

ITPP A scientific computing library. Used for its implementation of the Yule-Walker method for estimating filter coefficients for a given magnitude response.

JUCE Provides a framework for building graphical applications in C++. Used for the final application.

The project uses CMake to configure its build, and to automatically download project dependencies. Python and Octave were used for running and automating tests and generating graphs.

This documentation is written in Markdown, and compiled to html and to pdf using Pandoc. The project website is generated with Jekyll.

2 Image-source

Background

Basic Method

The image-source method aims to find the purely specular reflection paths between a source and a receiver. This relies on the simplifying assumption that sound propagates only along straight lines or "rays". Sound energy travels at a fixed speed, corresponding to the speed of sound, along these rays. The intensity of each "packet" of sound energy decreases with $1/r^2$, where r is the distance along the ray that the packet has travelled [31, p. 58].

Rays are perfectly reflected at boundaries. When a ray is reflected, it spawns a secondary source "behind" the boundary surface. This source is located on a line perpendicular to the wall, at the same distance from it as the original source, as if the original source has been "mirrored" in the surface. This is a first-order reflection. A ray which is reflected from several boundaries is represented by a "higher-order" image-source, which has been mirrored in each of those boundaries in turn [30, p. 104].

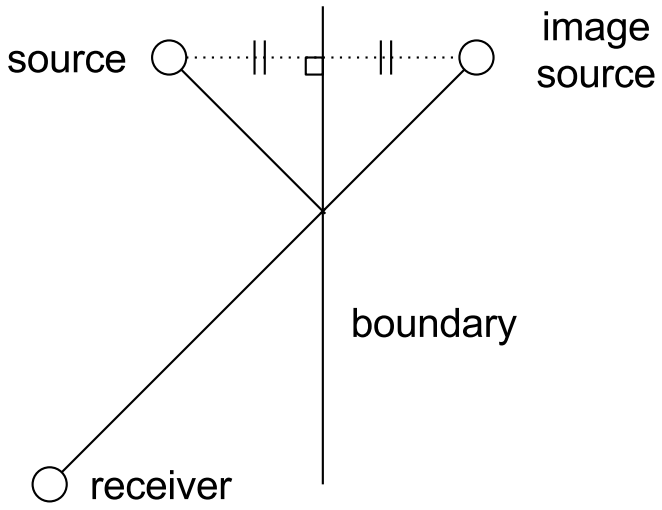

Figure 2.1: Image sources are found by reflecting the source position in a boundary.

All sources, original and image, emit the same impulsive source signal at the same time. The total impulse response (i.e. sound pressure against time) is found by summing the signals from each source, delayed and attenuated appropriately depending on the distance between that source and the receiver. The frequency response of the signal from each image source will additionally be modified depending on the characteristics of the boundaries in which that source was reflected.

Each image source represents a perfect specular reflection, so there is no way for the image model to calculate scattered or diffuse responses. Though this may sound troubling, it is not very problematic. The conversion of specular into diffuse sound energy is unidirectional, so repeated reflections cause the ratio of scattered to specular energy to increase monotonically. It is shown in [30, p. 126] that though the earliest reflections may be largely specular, after a few reflections the large majority of

sound energy becomes diffuse. This suggests that the image model should be used only for very early reflections, and a secondary model used to compute late, diffuse reflections.

Validity Checks

Having found the position of an image-source, by reflecting it in one or more surfaces, it must be checked to ensure it represents a specular path to the receiver. This is known as an *audibility test* [31, p. 202].

Consider first a source S, a receiver R, and a single wall A. The source is reflected in A, creating an image-source S_A. A line is constructed from R to S_A. If this line intersects A, then S_A represents a valid image source. Otherwise, there is no possible specular reflection involving S, R and A.

Now consider two walls, A and B. The image source S_{AB} has been reflected in A then B. For the image-source to be valid:

- $R \rightarrow S_{AB}$ must intersect B at some point $B_{\text{intersection}}$,
- $B_{\text{intersection}} \rightarrow S_A$ must intersect A at $A_{\text{intersection}}$, *and*
- $A_{\text{intersection}} \rightarrow S$ must not intersect with any scene geometry.

The validation of a third-order image-source will require three intersection checks, a fourth-order image will require four checks, and so on. This method of tracing backwards from the receiver to each of the image sources is known as *backtracking*.

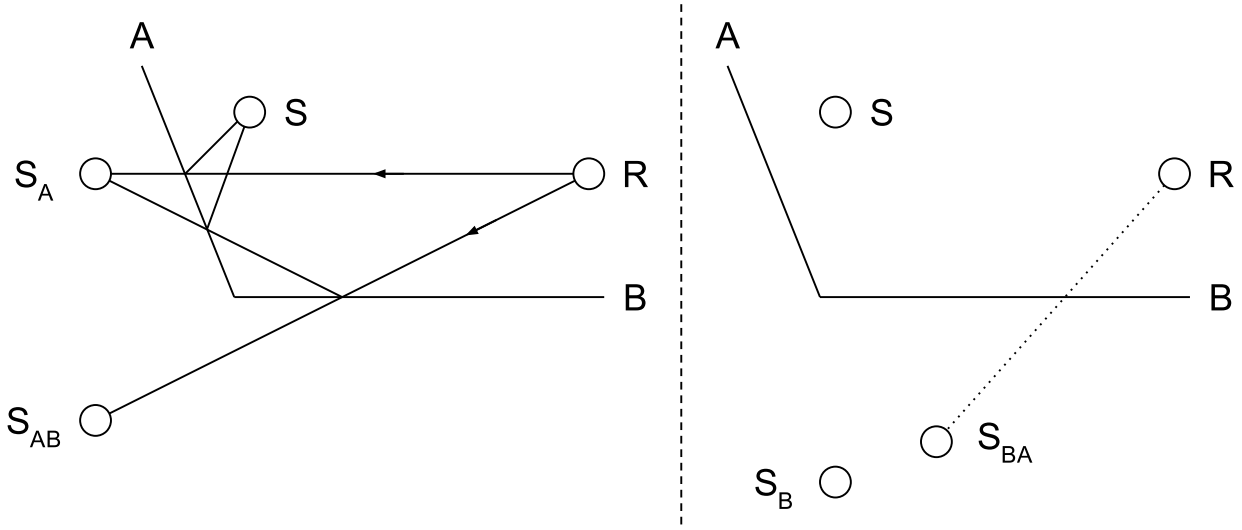

Figure 2.2: **Left:** The paths $S \rightarrow A \rightarrow R$ and $S \rightarrow A \rightarrow B \rightarrow R$ are both valid. **Right:** $S \rightarrow B \rightarrow A \rightarrow R$ is an invalid path because $R \rightarrow S_{BA}$ does not intersect A.

Acceleration

The naive method to find all the image sources for a scene is very expensive. Consider that to find a single first-order image source, the original source must be mirrored in a surface, and then an intersection test must be conducted between that surface and the image-source-to-receiver ray. To find all first-order image sources, this process must be carried out for all surfaces in the scene. To find all second-order image sources, each of those first-order images must be tested against every surface. This continues for higher-order images, so that the number of checks for image sources of a given order is equal to the number of surfaces raised to the power of that order. The relationship between the image-source order and the computation time is therefore exponential, meaning that high orders are impossible to compute in a reasonable time.

The majority of higher-order image sources found with the naive algorithm will be invalid. That is, they will fail the visibility/intersection test. For example, [30, p. 323] shows that, for tenth-order image-sources in a shoebox-shaped room, there are around 1.46e7 different image sources, only 1560 of which are valid. If the invalid image-sources can be discarded early, without requiring individual checking, then the amount of computation can be greatly reduced to a viable level. As explained above, image sources above order four or five are rarely required, but even these can be very time-consuming to find with the naive method. Optimisations are, therefore, a necessity for any but the simplest simulations.

To accelerate the image-source process, [31] suggests tracing a large number of rays in random directions from the source, and logging the unique paths of rays which eventually intersect with the receiver. Each unique path found in this way is used to generate an image source sequence, which is then checked as normal. This technique has the advantage that the majority of surface sequences are *not* checked, so the image-source process is fast. However, if the preliminary ray-tracer is not run with enough rays, it is likely to miss some valid paths, especially in complex scenes. Additionally, if the receiver volume is too great, then some invalid paths may still be detected.

The technique used by Wayverb is similar to that presented in [31], but makes a small change. A large number of random rays are traced, as before, but at each reflection point, the receiver is checked to see whether it is visible. If it is, then the surface sequence is checked for a valid image-source. This technique has two main advantages: more paths are checked, so it is more likely to find all the valid image-sources; and ray paths don't have to be specular, so ray-tracing can use techniques like *vector-based scattering*. The disadvantage is that a greater number of validity checks are required, though this number is still many times smaller than would be required by a naive implementation.

Implementation

Here the concrete implementation of the image-source method is presented, as it is used in Wayverb. Details of the microphone-modelling process are discussed separately, in the Microphone Modelling section. The following image (2.3) gives an overview of the entire process.

First, an axis-aligned bounding box is computed for the scene, and split into uniformly sized cuboid *voxels*. Each voxel holds a reference to any triangles in the scene which happen to intersect with that voxel. The voxel mesh acts as an "acceleration structure", speeding up intersection tests between rays and triangles. To check for an intersection between a ray and a number of triangles, the simplest method is to check the ray against each triangle individually, which is very time consuming. The voxel mesh allows the number to checks to be greatly reduced, by checking only triangles that are within voxels that the ray intersects. These voxels can be found very quickly, by "walking" the voxels along the ray, using an algorithm presented in [40]. For large scenes with many triangles, this method can lead to speed-ups of an order of magnitude or more. Assume all ray-intersection tests mentioned throughout this thesis use the voxel-acceleration method, unless explicitly noted.

Rays are fired in uniform random directions from the source. Each ray is checked for an intersection with the scene, and if an intersection is found, some data about the intersection is recorded. Specifically, the record includes the triangle which was intersected, and whether or not the receiver is visible from the intersection point. Then, the vector-based scattering method [41] is used to find the directions of new rays, which are fired from the intersection points. The ray-tracing process continues up to a certain depth, which is artificially limited to ten reflections in Wayverb. For most simulations, three or four reflections should be adequate, though this depends somewhat on the scattering coefficients of the surfaces, as explained in the Basic Method subsection.

The ray tracer produces a list of reflection paths for each ray. Some rays may follow the same paths, and so duplicates must be removed. This is achieved by condensing per-ray information into a tree of valid paths. Each node in the tree stores a reference to a triangle in the scene, and whether or not the receiver is visible from this triangle. Each unique path starting from a root node in the tree represents

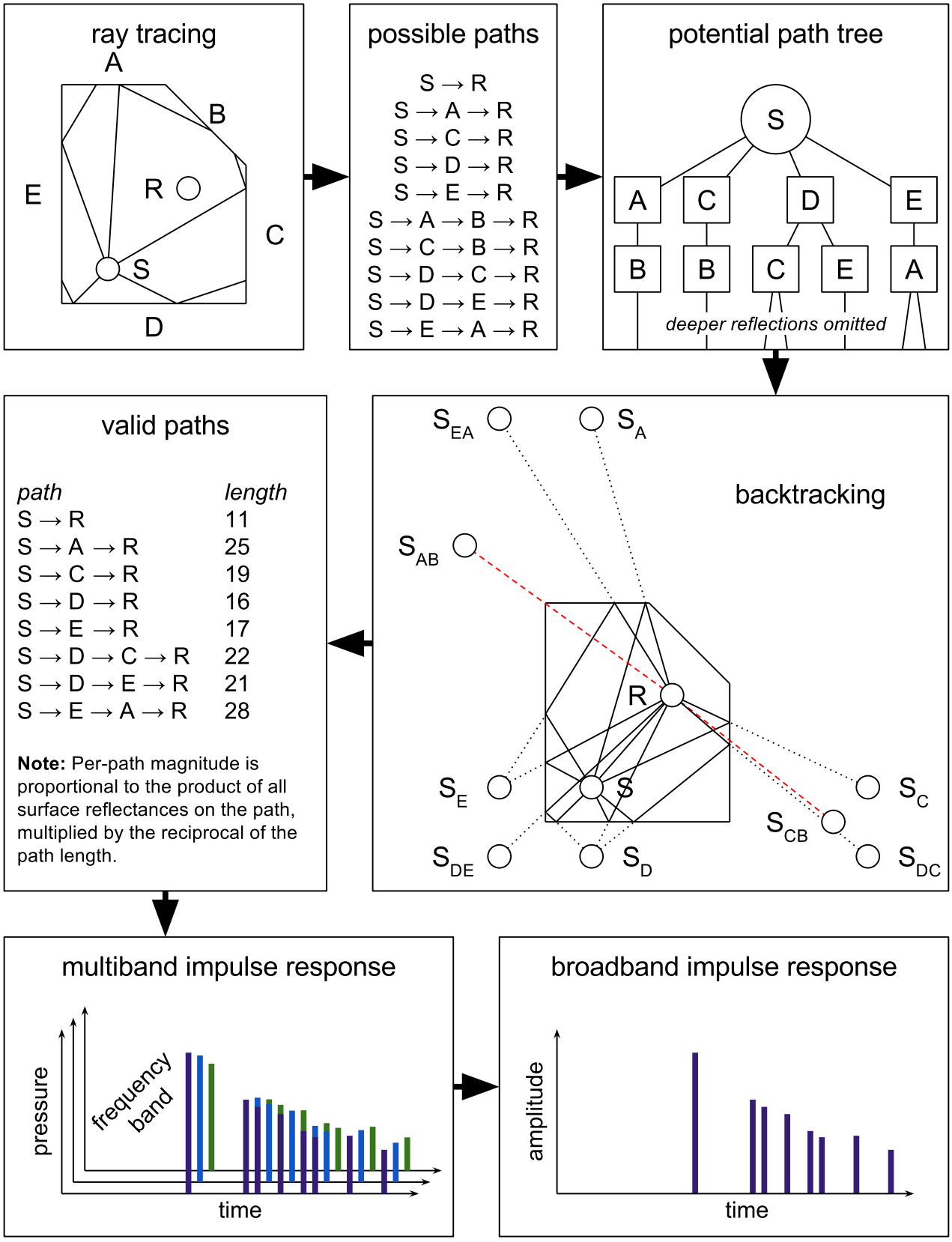

Figure 2.3: Creation of an impulse response using image sources.

a possible image source contribution, which must be checked. This checking is carried out using the backtracking method explained above. A nice property of the tree structure is that it can be traversed using depth-first recursion, allowing the results of some intermediate calculations to be cached between ray paths, speeding up the calculation with only minimal memory overhead. This is similar to the approach mentioned in [2]. Also, because the tree is immutable, it can be shared between multiple worker threads, which independently check each branch for valid image sources. The nature of the recursive algorithm makes it a poor fit for an OpenCL implementation, so native (CPU) threads are used instead.

Some paths in the tree may not actually produce valid image sources, and these paths are discarded. For paths which *do* contribute valid image sources, the propagation delay and frequency-dependent pressure of the image source signal must be found. According to [30, p. 325], the propagation delay is equal to the distance from the receiver to the image source, divided by the speed of sound. The pressure content is found by convolving together the reflectances of all intermediate surfaces. This is equivalent to a single multiplication per frequency band, as long as reflectances can be represented by real values.

The surface reflectances are found by converting per-band absorptions into per-band normal-incidence reflectance magnitudes by $|R| = \sqrt{1-\alpha}$. These are converted to per-band impedances by

(1)
$$\xi = \frac{1+|R|}{1-|R|}$$

Finally, the impedances are converted back to *angle-dependent* reflectances by

(2)
$$R(\theta) = \frac{\xi\cos\theta - 1}{\xi\cos\theta + 1}$$

where θ is the angle of incidence at the surface. This is the same approach taken in [27].

The contribution g of a single image source with intermediate surfaces $m_1 m_2 \dots m_n$ is given by

(3)
$$g_{m_1 m_2 \dots m_n} = \frac{\sqrt{Z_0/4\pi}}{d_{m_1 m_2 \dots m_n}} \cdot r_{m_1} * r_{m_2} * \cdots * r_{m_n} * \delta(\frac{d_{m_1 m_2 \dots m_n}}{c})$$

where Z_0 is the acoustic impedance of air, c is the speed of sound, $d_{m_1 m_2 \dots m_n}$ is the distance from the receiver to the image source, and r_{m_i} is the reflectance of surface i. This assumes that the original source emits a pressure impulse δ at the starting-time of the simulation. The contributions of all image sources must be summed together to find the final impulse response.

To create a digital audio file representing an impulse response, the output must be discretised at some sampling frequency f_s. The individual image source contributions must be added, at positions corresponding to their propagation delays, into an output buffer at that sampling frequency. The ideal buffer position for a given contribution is equal to τf_s where τ is the propagation delay of that contribution. However, this value is unlikely to be an integer, and so will not coincide with a sample index. The simplest solution would be to round to the closest integer, and use this as the sample index. However, for applications such as multi-microphone simulation which are sensitive to arrival time, this can lead to obvious phase errors. A better solution is suggested in [42]: The contribution can be positioned with sub-sample accuracy, by replacing the impulsive δ signal with the impulse-response of an ideal low-pass filter, with cut-off equal to the output Nyquist frequency. Such an impulse response is infinitely long, but tends to zero quickly, so it can be Hanning windowed to reduce the number of additions required. This form of the impulse is as follows:

(4)
$$\delta_{\mathrm{LPF}}(n-\epsilon) = \begin{cases} \frac{1}{2}(1+\cos\frac{2\pi(n-\epsilon)}{N_w})\mathrm{sinc}(n-\epsilon), & -\frac{N_w}{2} < n < \frac{N_w}{2} \\ 0, & \text{otherwise} \end{cases}$$

where n is an index in the output buffer, ϵ is the centre of the impulse in samples ($\epsilon = \tau f_s$), and N_w is the width of the window in samples.

Recall that each image-source contribution has per-band pressure values. Rather than summing all contributions directly to the output buffer, several buffers are created, one per frequency band. The contributions for each band are summed into each buffer individually. The final output of the simulation is created by band-passing and then mixing down the buffers.

Integration with Ray Tracing Algorithm

The beginning of the image-source process relies on randomly ray tracing a certain number of reflections. This ray tracing process is similar to that used for estimating late, diffuse reflections. When the simulation is run, rays are actually traced to a much greater depth of maybe 100 reflections or more. The first few reflections are routed to image-source processing, while the entire set of reflections is used for finding the reverb tail.

It is important to note that the stochastic ray tracing process will record both specular and diffuse reflections. At the beginning of the impulse response, this will lead to a duplication of energy, as the energy from specular reflections will be recorded by both the image-source and ray-tracing processes. To solve this problem, the stochastic ray tracer records specular and diffuse contributions separately. Specular contributions from the ray tracer are only added to the output for reflections of higher order than the highest image-source order.

A second problem is surface scattering. When simulating scenes with high surface scattering coefficients, specular reflections should be quiet, with a greater degree of scattered energy. Unfortunately, the image-source process cannot account for scattered sound energy by design. The solution is to use diffuse contributions from the stochastic ray tracer, so that the image-source and ray-traced outputs "overlap". To ensure that the amount of energy in the simulation remains constant, the image-source finder must account for energy lost to scattering during reflections. Therefore, after finding the reflectance of each surface using the method outlined above, the reflectance is further multiplied by $(1 - s)$ where s is the frequency-dependent scattering coefficient of the surface. This causes the image-source contributions to die away faster, and the "missing" energy will be made up by the diffuse output of the ray tracer.

3 Ray Tracer

Background

Similarly to the image-source method, ray tracing assumes that sound energy is transported around a scene in "rays". The rays start at the sound source, and are emitted at the same time, travelling at the speed of sound. When a ray hits a boundary, it loses some energy, and is reflected. When it intersects the receiver, the energy and time-delay of the ray is recorded. In these ways, the models are similar. However, there are some important differences between the two methods, explained below.

Stochastic Simulation

Image sources are deterministic, while ray tracing is stochastic. The image-source method finds exact specular reflections, which will be constant for given source, receiver, and boundary positions. Ray tracing is less accurate, aiming to compute a result which is correct, within a certain probability. A large number of rays are fired into the scene in random directions, and reflected up to a certain depth. Some of these rays may intersect with the receiver volume, but some may not. Only rays that *do* intersect the receiver contribute to the final output. The proportion of rays which intersect the receiver, and therefore the measured energy at the receiver, can be found with greater accuracy simply by increasing the number of rays fired.

Receiver Volume

The random nature of ray tracing requires that the receiver must have a finite volume. The likelihood of any given ray intersecting with a single point with no volume is zero. If the probability of a ray-receiver intersection is to be non-zero, the receiver must have some volume. This is different to the image-source method, which traces reflections backwards from the receiver, allowing it to be represented as a point.

Energy and Distance

In ray tracing, each ray represents a finite portion of the initial source energy. The reduction of energy over a given distance is accounted-for by the spreading-out of the rays. This can be illustrated very simply: First, imagine a sphere placed very close to a point. Assuming rays are fired with a uniform random distribution from that point, a certain proportion of those rays will intersect with the sphere. If the sphere is moved further away, a smaller proportion of rays will hit it (see the following figure (3.1)).

The exact proportion of intersecting rays is equal to $s/4r^2$ [32, p. 75], where s is the constant area covered by the receiver, and r is the distance between the source and receiver. That is, the proportion of rays intersecting the receiver is inversely proportional to the square of the distance between the source and receiver. The energy recorded is proportional to the number of ray intersections recorded, therefore, the ray model intrinsically accounts for the inverse-square law for energy, and the per-ray energy does not need to be scaled proportionally to the distance travelled. This differs to the image-source model, in which only valid specular reflections are recorded, and the inverse-square law may be applied directly.

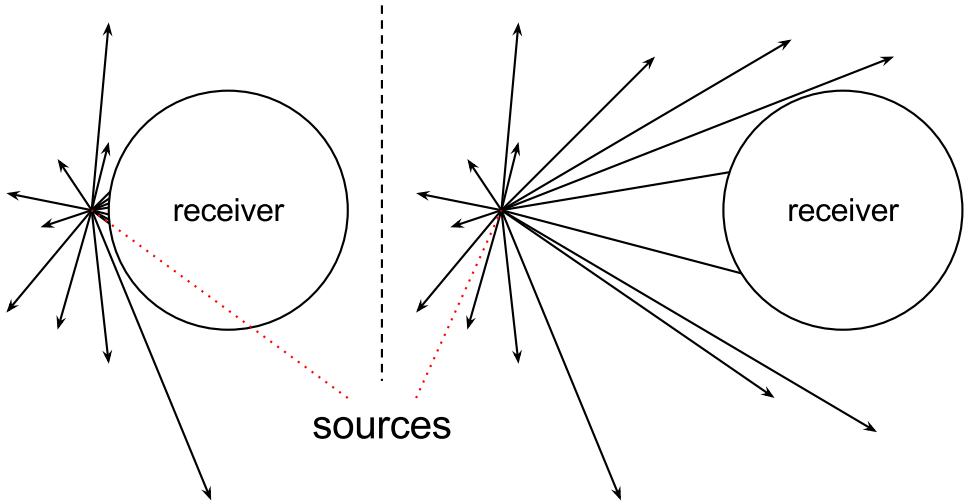

Figure 3.1: The proportion of randomly-distributed rays intersecting with a sphere depends on the distance between the ray source and the sphere.

Rendering

The final major difference between ray tracing and the image-source method is to do with the way in which results are recorded. The image-source method finds exact specular reflections, each of which contributes an impulsive signal with specific frequency content at a precise time. This reflection data is precise and accurate, so it can be used to render an output signal at arbitrarily high sampling frequencies. Ray tracing, on the other hand, is inexact, based on statistical methods. For a given unit time, the number of rays detected must be very high in order for the detected energy level to be accurate within certain bounds. It is shown in [31, p. 191] that the mean number of intersections k per time period Δt is given by

(5)
$$k = \frac{N\pi r^2 c\Delta t}{V}$$

where N is the number of rays, r is the radius of the receiver, c is the speed of sound, and V is the room volume.

For an output which covers the human hearing range, the sampling rate must be at least 40KHz, which corresponds to a sampling period of 25s. Therefore, for a receiver radius of 0.1m, and assuming 100 detections-per-second is adequate, the minimum number of rays is

(6)
$$N = \frac{kV}{\pi r^2 c\Delta t} = \frac{100V}{340 \cdot 0.000025 \cdot 0.1^2\pi} \approx 374500V$$

For higher accuracy, higher output sample rates, and smaller receivers the number of rays required becomes even greater. Even on modern hardware, this sheer quantity of rays is prohibitive.

If, on the other hand, audio-rate results are not required, then the number of necessary rays is much lower. [31, p. 186] suggests a sampling period of the order of magnitude of milliseconds, which requires at least 40-times fewer rays.

Now, the ray tracer can be thought to produce an *energy envelope*, describing the decay tail of the impulse response. To produce the impulse response itself, this energy envelope is simply overlaid onto a noise-like signal. The process will be described in greater detail in the following Implementation section.

Implementation

Here, Wayverb's ray tracer will be described. Details of the boundary- and microphone-modelling processes are discussed separately, in the Boundary Modelling and Microphone Modelling sections respectively.

Finding Reflections

The simulation begins identically to the image-source process. A voxel-based acceleration structure is created, to speed up ray intersection tests.

Rays are fired in uniformly-distributed random directions from the source point. Each ray carries a certain quantity of energy (the method for determining the starting energy is described in the Hybrid section). If a ray intersects with the scene geometry, some data is stored about that intersection: its position, the unique ID of the triangle which was intersected, and whether or not the receiver point is visible from this position. This data will be used later on, when calculating energy loss, and the directional distribution of received energy.

A new ray direction is calculated using the *vector-based scattering* method, described in [41]. A uniformly random vector is generated, within the hemisphere oriented in the same direction as the triangle normal. The ideal specular direction is also calculated, and the two vectors are combined by

(7)
$$\overrightarrow{R}_{\text{outgoing}} = s\overrightarrow{R}_{\text{random}} + (1-s)\overrightarrow{R}_{\text{specular}}$$

where s is the scattering coefficient. Normally, the scattering coefficient would be defined per-band, but this would require running the ray tracer once per band, so that each frequency component can be scattered differently. Instead, the mean scattering coefficient is used, so that all bands can be traced in one pass.

Having calculated a new ray direction, the energy carried in the ray is decreased, depending on the absorption coefficients of the intersected triangle. If the surface has per-band absorptions coefficients α, then the energy in each band is multiplied by $(1-\alpha)$ to find the outgoing energy. The new ray, with the computed outgoing energy and vector-scattered direction, is now traced.

The ray tracing process continues for a set number of reflections. Typically, each ray would be traced until the energy in all bands has fallen below a certain threshold, requiring an additional check per reflection per ray [31, p. 183]. Under such a scheme, some rays might reach this threshold faster than others, depending on the absorptions of intermediate materials. However, in Wayverb all rays are traced in parallel, so it is not feasible or necessary to allow rays to quit early. Instead, the maximum possible required depth is found before the simulation, and all rays are traced to this maximum depth.

To find the maximum ray tracing depth, first the minimum absorption of all surfaces in the scene is found. The outgoing energy from a reflection is equal to $E_{\text{incoming}}(1-\alpha)$ where E_{incoming} is the incoming energy and α is the surface absorption. The maximum ray tracing depth is equal to the number of reflections from the minimally absorptive surface required to reduce the energy of a ray by 60dB:

(8)
$$n_{\text{reflections}} = \left\lceil -\frac{6}{\log_{10}(1-\alpha_{\text{min}})} \right\rceil$$

Logging Energy

The output of the ray tracing process is a histogram, plotting recorded energy per time step. This recorded energy may come from two different sources.

Firstly, if a ray intersects with the receiver volume, then the current energy of that ray, which may have been attenuated by previous reflections, is added to the histogram at the appropriate time step. The time of the energy contribution is given by the total distance travelled by the ray, divided by the speed of sound. This is the approach taken in typical acoustic ray tracers.

Secondly, each reflection point is considered to spawn a "secondary source" which emits scattered sound energy. If the receiver is visible from the reflection point, then a small energy contribution is logged, at a time proportional to the distance travelled by the ray. The exact level of this contribution is explained in the Geometric Implementation subsection of the Boundary Modelling page.

Producing Audio-rate Results

When ray tracing has completed, the result is a set of histograms which describe the energy decay envelope of each frequency band. These histograms will have a relatively low sampling rate of around 1KHz, as explained above, so they are not directly suitable for auralisation. To produce audio-rate impulse responses, the "fine structure" of the decay tail must be synthesised and then enveloped using the histogram envelopes. The process used to convert the histogram into an audio-rate impulse response is described in [43], and in greater depth in [32, p. 70], though an overview will be given here. The following image (3.2) outlines the process of estimating an audio-rate representation of low-sample-rate multi-band histograms.

Generating a Noise Signal

First, a noise-like sequence of Dirac impulses is generated, at audio-rate. This sequence is designed to mimic the density of reflections in an impulse response of a certain volume. Therefore it starts sparse, and the density of impulses increases with time. Specifically, the time between one impulse event and the next is given by

(9)
$$\Delta t_{\text{event}}(z) = \frac{\ln \frac{1}{z}}{\mu}$$

where z is a uniformly distributed random number $0 < z \leq 1$. μ here is the mean event occurrence, and is dependent upon the current simulation time t, the enclosure volume V and the speed of sound c:

(10)
$$\mu = \frac{4\pi c^3 t^2}{V}$$

It can be seen that the mean occurrence is proportional to the square-inverse of the current time, producing an increase in event density over time. The first event occurs at time t_0:

(11)
$$t_0 = \sqrt[3]{\frac{2V \ln 2}{4\pi c^3}}$$

The full-length noise signal is produced by repeatedly generating inter-event times Δt_{event}, and adding Dirac impulses to a buffer, until the final event time is greater or equal to the time of the final histogram interval. Dirac deltas falling on the latter half of a sampling interval are taken to be negative-valued. The number of Dirac deltas per sample is limited to one, and the value of μ is limited to a maximum of 10KHz, which has been shown to produce results absent of obvious artefacts [43].

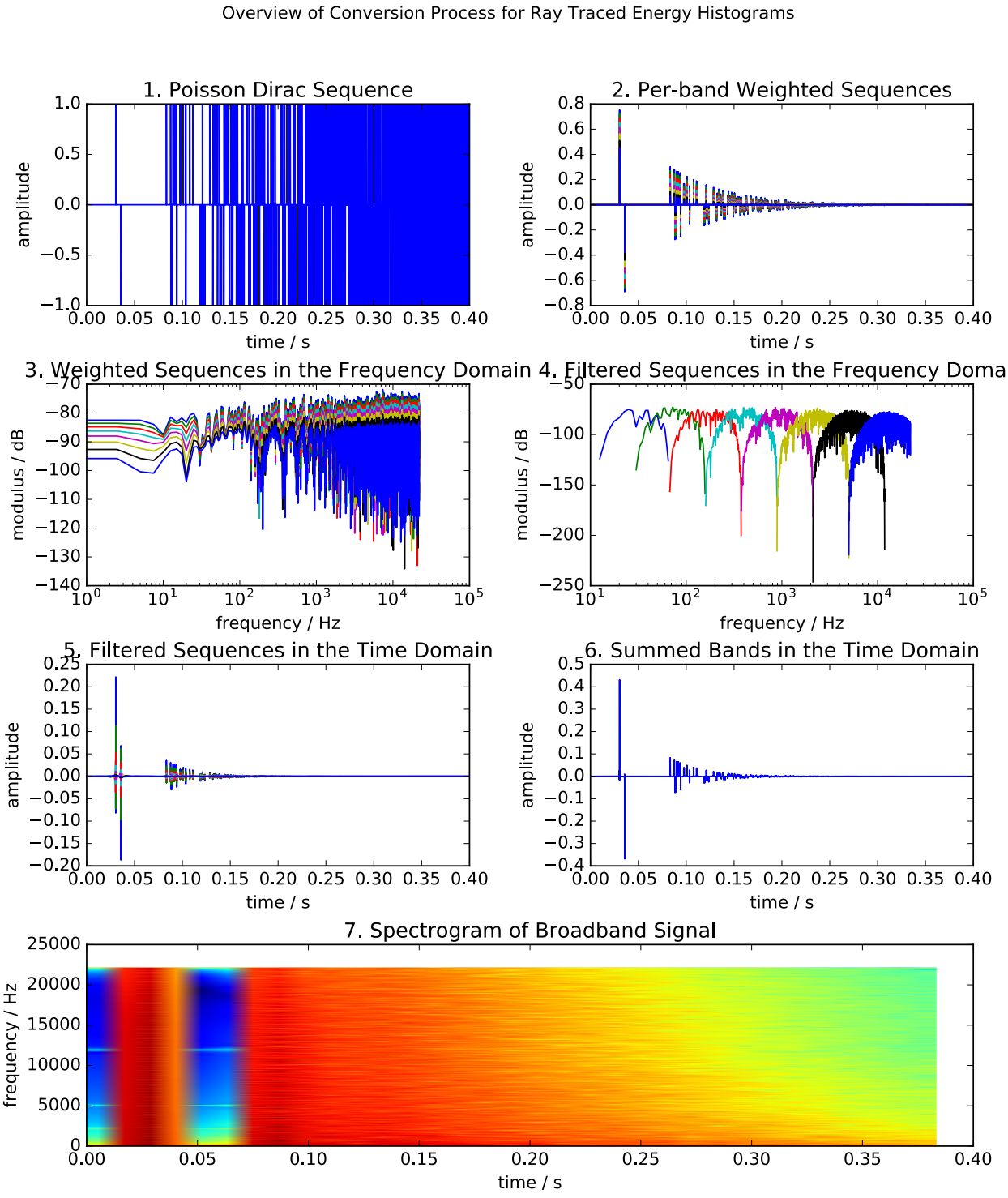

Figure 3.2: Generating an audio-rate signal from multi-band ray tracing energy histograms at a low sampling rate.

Weighting Noise Sequences

The noise sequence is duplicated, once for each band. Then, the noise sequence for each band is weighted according to that band's histogram. This enveloping is not quite as simple as multiplying each noise sample with the histogram entry at the corresponding time. Instead, the enveloping process must conserve the energy level recorded over each time step.

For each time interval in the histogram, the corresponding range of samples in the noise sequence is found. If the output sample rate is f_s and the histogram time step is Δt, then the noise sequence sample corresponding to histogram step h is $\lfloor h \cdot f_s \cdot \Delta t \rfloor$. The corrected energy level for each histogram step is found by dividing the histogram energy value by the sum of squared noise samples for that step. This is converted to a corrected pressure level by $P = \sqrt{Z_0 I}$, where I is the corrected energy level, and Z_0 is the acoustic impedance of air. The weighting is now accomplished by multiplying each noise sequence sample corresponding to this histogram step by the corrected pressure level.

Multi-band Filtering

Now, we are left with a set of broadband signals, each with different envelopes. The output signal is found by bandpass filtering each of these signals, and then mixing them down.

The filtering of each signal is accomplished by computing the signal's frequency-domain representation, attenuating bins outside the passband, and then transforming the altered spectrum back to the time domain. To ensure perfect reconstruction, and to avoid artificial-sounding discontinuities in the spectrum, the filter shape suggested in [44] is used. This paper suggests equations which describe the band-edge magnitudes:

(12)
$$G_{\text{lower}}(\omega_{\text{edge}} + p) = \sin^2\left(\frac{\pi}{2}\phi_l(p)\right), G_{\text{upper}}(\omega_{\text{edge}} + p) = \cos^2\left(\frac{\pi}{2}\phi_l(p)\right)$$

Here, G is a function of frequency, ω_{edge} is the band-edge frequency, and p is the relative frequency of a nearby frequency bin. The equations are computed for a range of values $p = P, \ldots, P$ where P is the width of the crossover. The definition of $\phi_l(p), l \geq 0$ is recursive:

(13)
$$\phi_l(p) = \begin{cases} \frac{1}{2}(p/P + 1), & l = 0 \\ \sin(\frac{\pi}{2}\phi_{l-1}(p)), & \text{otherwise} \end{cases}$$

The variable l defines the steepness of the crossover, and is set to 0 in Wayverb, so that the transition between bands is smooth and gradual. The absolute width of the crossover is denoted by P, but it is more useful to specify the crossover width in terms of overlap $0 \leq o \leq 1$. Assuming logarithmically-spaced frequency bands, spread over the range $\omega_{\text{lowest}}, \ldots, \omega_{\text{highest}}$ where the edge frequency of band i is defined as

(14)
$$\omega_{\text{edge}_i} = \omega_{\text{lowest}}\left(\frac{\omega_{\text{highest}}}{\omega_{\text{lowest}}}^{\frac{i}{N_{\text{bands}}}}\right)$$

the maximum width factor w is given by

(15)
$$w = \frac{x-1}{x+1}, x = \frac{\omega_{\text{highest}}}{\omega_{\text{lowest}}}^{\frac{1}{N_{\text{bands}}}}$$

For $\omega_{\text{lowest}} = 20\text{Hz}$, $\omega_{\text{highest}} = 20\text{KHz}$, and $N_{\text{bands}} = 8$, $w \approx 0.4068$. Then, the band edge width P can be defined in terms of the overlap-amount o, the frequency of this band edge ω_{edge}, and the maximum allowable width factor w: $P = \omega_{\text{edge}} o w$. Wayverb sets the overlap factor $o = 1$ to ensure wide, natural-sounding crossovers.

The final broadband signal is found by summing together the weighted, filtered noise sequences.

4 Digital Waveguide Mesh

Background

The *digital waveguide mesh* (DWM) is one of several wave-based simulation techniques. Each technique in this family is derived directly from the wave equation, allowing them to inherently support wave phenomena such as diffraction and interference. Wave effects such as these are particularly important to the low-frequency response of a room. This means that at low frequencies wave-based methods are far more accurate than geometric methods, which are not able to model wave effects [1].

The drawback of wave-based methods is that their computational complexity increases rapidly with the maximum output frequency, and with the size of the modelled space. On current consumer hardware, it is not feasible to compute a full-spectrum simulation using wave-based techniques. To optimise computation time, while retaining reasonable accuracy across the spectrum, wave-based methods can be combined with geometric methods. Wave-based methods are used to calculate accurate low-frequency content, while geometric methods can estimate the higher frequency content, which is less dependent upon wave effects.

There are, largely speaking, two main types of wave-based simulation used for room acoustics: element methods, and finite difference methods. The waveguide mesh is the latter, a simplified sub-class of the *finite-difference time-domain* (FDTD) technique. Although the DWM and FDTD have converged over time, and are equivalent [45], [46], their histories are quite different. The DWM was designed to be efficient for small-scale acoustic simulations in which the only quantity of interest is pressure [3], while FDTD is a more general technique designed for electromagnetic simulation, in which the electric and magnetic fields are both of interest [47].

Method

The derivation of the waveguide mesh begins with the one-dimensional *digital waveguide*. A one-dimensional waveguide can exactly describe the behaviour of a band-limited wave in one dimension. Such a model is well-suited for predicting the behaviour of certain musical instruments which use columns of air or vibrating strings to produce sound. The model itself is based on d'Alembert's solution of the wave equation in one dimension [5, p. 86]. The displacement of a string y at time t and position x can be written as

(16)
$$y(t,x) = y_r\left(t - \frac{x}{c}\right) + y_l\left(t + \frac{x}{c}\right)$$

where $y_r\left(t - \frac{x}{c}\right)$ and $y_l\left(t + \frac{x}{c}\right)$ are the right- and left-going travelling waves respectively, with speed c. The discrete form of this equation is given in terms of constant time and space divisions T and X:

(17)
$$y(nT, mX) \triangleq y^+(n-m) + y^-(n+m)$$

where superscript $+$ and $-$ denote propagation to the right and left respectively, and n and m are integral [45].

An implementation of these equations will take the form of two parallel delay lines, which propagate wave components in opposite directions. This is shown in the following diagram (4.1). The "output" of

the simulation, that is, the physical displacement of the modelled string over time, is found by adding the wave components in both delay lines at a single point.

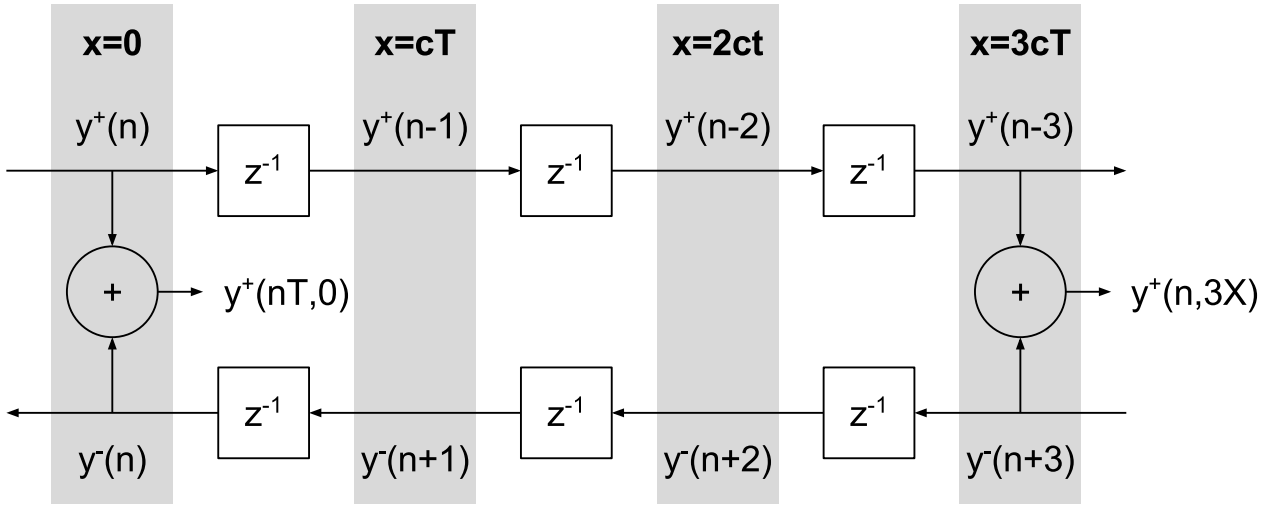

Figure 4.1: Delay lines cause wave components to be propagated along the "string" over time. The total displacement of the string is found by adding together values from the same point on each delay line.

Waveguides in higher dimensions can be created in a straightforward manner, by connecting digital waveguide elements at a *scattering junction*. Wave components entering the junction are distributed amongst the waveguide elements, preserving energy and power according to Kirchoff's power conservation laws [5, p. 87]. The sound pressure p_J at a lossless scattering junction J with N connected elements or "ports" is the summed incoming components of all connected elements:

(18)
$$p_J = \frac{2\sum_{i=1}^{N} \frac{p_i^+}{Z_i}}{\sum_{i=1}^{N} \frac{1}{Z_i}}$$

where p_i is the pressure in waveguide element i and Z_i is its associated impedance. This simplifies, if all impedances are equal, which is true for homogeneous media:

(19)
$$p_J = \frac{2}{N}\sum_{i=1}^{N} p_i^+$$

A *digital waveguide mesh* is any configuration of regularly-arranged N-port scattering junctions which are separated by unit delay lines. In some literature, this specific type of mesh is known as a *W-DWM* because it operates in terms of "W" (wave) variables.

The input into a scattering junction is equal to the output of a neighbour junction at the previous time step. This fact allows the waveguide mesh to alternatively be formulated directly in terms of the pressure at each junction (assuming all junction impedances are equal) [18]:

(20)
$$p_J(n) = \frac{2}{N}\sum_{i=1}^{N} p_i(n-1) - p_J(n-2)$$

That is, the next pressure at a given node depends on the previous pressure at that node, and the current pressure at surrounding nodes. This alternative formulation operates on Kirchhoff variables, and is therefore known as a *K-DWM*. In the 1D case, the K-DWM and W-DWM are computationally identical [45]. In higher dimensions, they are equivalent only under certain circumstances [48, p. 5].

The K-DWM is advantageous compared to the W-DWM for reasons of efficiency. It requires less memory, and fewer calculations per node per step than the W-DWM: experiments in [18] show that the K-DWM is 200% faster and uses 50% of the memory of the equivalent W-DWM. This is mainly to do with the number of values stored per node. Each node in a K-DWM must store a "current" and a "previous" pressure value, whereas in a W-DWM each *connection* must store a value [46]. For mesh layouts in which each node has many neighbours (see Mesh Topology below), the W-DWM can require many times more memory. The K-DWM also requires one fewer addition per node per step [5, p. 91], so will be slightly faster, all else being equal.

In the general case of an N-dimensional waveguide mesh, the spatial and temporal sampling periods are related by the Courant number λ. The Courant criterion specifies the conditions required for numerical stability of the simulation:

(21)
$$\lambda = \frac{cT}{X} \leq \frac{1}{\sqrt{N}}$$

The highest sampling rate and lowest error is achieved by setting the Courant to its maximum value [49]. This is normally desirable, and so the inequality above can be simplified:

(22)
$$T = \frac{X}{c\sqrt{N}}$$

Note that a higher output sampling rate requires a smaller inter-nodal spacing and therefore more modelled points per-unit-volume, which in turn requires more memory and more calculations per time step. Unfortunately, the valid bandwidth of the output is generally smaller than the maximum available bandwidth, i.e. up to the Nyquist frequency [36]. The waveguide sampling rate must generally be at least four times the maximum required output frequency for higher-dimensional simulations (more detailed bandwidths for some common topologies are given in [36]).

Strengths and Weaknesses of the DWM

The main advantage of the DWM is its relative simplicity. The air in the simulation is evenly divided into nodes. Each node has an associated pressure, and also stores its previous pressure. The next pressure at a node depends on its previous pressure, and the current pressures at its neighbours. The simulation progresses by repeatedly updating all nodes, and storing the change in pressure over time at some output nodes.

This simplicity presents an optimisation opportunity. Each node can be updated completely independently, as long as all updates in one time-step are completed before the next time-step begins. This means that the updates can happen in parallel. In fact, with enough memory and processing cores, the entire mesh could be updated in the time that it takes to update a single node. The update method is also very simple, only requiring some additions and a multiplication per node per step. This kind of simple, parallel code lends itself to implementations on graphics hardware, which is designed for running similar calculations simultaneously over large inputs. *Graphics processing units* (GPUs) have best throughput when all threads execute the same instruction, and when memory is accessed at consecutive addresses from running threads. The K-DWM update equation has no branching, and has consistent memory access patterns, which should allow for a very efficient implementation for GPUs.

The greatest limitation of the DWM is *dispersion error*. Unlike waves in homogeneous physical media, the velocity of wave propagation in the DWM depends on the direction of propagation, and also on the frequency of the wave component. This leads to errors in the frequency response of recorded signals, especially toward the upper limit of the output bandwidth. The exact pattern of dispersion error is dependent upon the topology of the mesh, and can be examined using *Von Neumann* analysis [34]. One solution to the dispersion problem is to increase the sampling rate of the mesh, moving the high-error area out of the region of interest. Of course, this can quickly become very expensive, as the number of

nodes, and therefore memory usage and computation time is proportional to the inverse cube of the sampling period. Another option is to use a mesh topology designed to reduce both direction- and frequency-dependent error, such as those presented in [36]. One interesting variation on this option is to reduce only direction-dependent error, and then to compensate for frequency-dependent error with a post-processing step, which is the approach taken in [11]. However, these mesh topologies with higher accuracy and isotropy require relatively high numbers of calculations per node. The interpolated schemes in [36] require 27 additions and 4 multiplications per node, whereas a tetrahedral mesh would require 5 additions and 1 multiplication. It is clear that high-accuracy results will be very costly to compute, whichever method is used.

Design Choices

Mesh Type

Mesh Topology

There is no single optimal implementation of the digital waveguide mesh. Perhaps the most important decision is the mesh topology or *stencil* that will be used, and by extension the mesh update equation. Here, the mesh topology refers to the pattern which is used to distribute nodes throughout the modelled space. The choice of topology will affect the accuracy, memory usage, calculation speed, and implementation complexity of the final design. It must therefore be chosen with care, in order to satisfy the constraints of the particular application.

The simplest topology is rectilinear, in which nodes are laid out on the vertices of a cubic grid, and each node has 6 direct neighbours. During the mesh update, the pressure at each neighbour node must be checked in order to calculate the next pressure at the current node. This is straightforward to implement, as the nodes can be stored in memory in a three-dimensional array, in which the array extents define the mesh dimensions, and the array indices refer to the positions of individual nodes. Other options for the topology include tetrahedral, octahedral, and dodecahedral, in which nodes have 4, 8, and 12 neighbours respectively.

TODO topology diagrams

The accuracy may be increased by overlaying or "superposing" cubic, octahedral, and dodecahedral schemes together, as all nodes are oriented uniformly, and have cubic tessellation. Such schemes are known as *interpolated*, and in these schemes each node has 26 neighbours. The cubic, octahedral, dodecahedral, and interpolated schemes may additionally all be represented by a single "unified" update equation, described in [36]. In this respect the tetrahedral scheme is unique, requiring a dedicated update method. This is because the nodes in a tetrahedral mesh may be oriented in either of two directions, effectively requiring two update equations instead of one.

If the primary concern is speed rather than accuracy, a scheme with fewer neighbour nodes should be used, as the number of calculations per node is proportional to the number of neighbours [8]. This is especially true for the tetrahedral topology, which is the least dense. That is, it requires the fewest nodes to fill a volume at any given sampling rate. Fewer nodes to update means fewer calculations, and a faster overall simulation. Lower density meshes also require less storage, so the tetrahedral scheme is the most memory efficient.

Now, to optimise for accuracy, it now appears that there are two possible approaches: The first option is to use an interpolated scheme, which is relatively inefficient in terms of time and space requirements, but which is most accurate for any given sampling rate. The second option is to use a tetrahedral mesh, which is inaccurate but the most time- and space-efficient, and to oversample until the required accuracy is achieved. Unfortunately, there is no prior research comparing the accuracy and efficiency of the tetrahedral topology against interpolated schemes. Due to time constraints, this could not be investigated as part of the Wayverb project (implementation and numerical analysis of

mesh topologies is not trivial). It was, however, noted that the tetrahedral mesh is the more flexible of the two approaches. That is, when accuracy is not important, the tetrahedral mesh will always be most efficient, but it can be made more accurate by oversampling. On the other hand, the interpolated schemes cannot be tuned to produce less accurate results quickly - they will always be accurate but inefficient. For these reasons, the tetrahedral mesh was initially chosen for use in Wayverb.

The tetrahedral mesh was implemented during within the first two months of the project, with support for microphone modelling. When it came to implementing frequency dependent boundary conditions, no prior research could be found discussing boundary implementations in a tetrahedral topology. As noted in the conclusion of [36]:

> One aspect that has not been dealt with in this paper, and could be of interest for future research, is the comparison of the identified schemes with the tetrahedral topology… However, the applicability of the tetrahedral stencil to room acoustic simulations is debatable due to the nontrivial formulation of boundary conditions for complex room shapes.

The design of a new boundary formulation is outside the scope of this research, which was primarily concerned with the implementation of existing techniques rather than the derivation of new ones. Instead, the standard leapfrog (rectilinear) scheme was used. Most of the tetrahedral code had to be rewritten, as the update schemes are very different (tetrahedral nodes have two possible update equations depending on node orientation), and the memory layout and indexing methods are more involved. The new scheme is much simpler than the tetrahedral mesh, and was quick to implement.

The rectilinear mesh uses the same cubic tessellation as the more complex topologies mentioned earlier, so in the future the update equation could conceivably be replaced with a more accurate "interpolated" alternative. Such a scheme would yield better computational efficiency for simulations where high accuracy is required. The conversion would only require changes to the update equations (and would complicate the boundary modelling code), but would be more straightforward than the move from the tetrahedral to the rectilinear topology. Due to time constraints, this was not possible during the project, meaning that the waveguide in Wayverb is not the most optimal in terms of accuracy *or* speed - instead, it was optimised for ease and speed of implementation. Use of a more suitable mesh topology would be a sensible starting point for future development work on the project.

Source Excitation Method

The question of input and output in the digital waveguide mesh is superficially simple. The waveguide mesh is a physical model, and so it is easy to draw an analogy between the waveguide process, and the process of recording a physical impulse response. For physical spaces, a speaker plays a signal at some location within the space, and a microphone reads the change in air pressure over time at another point in the space. The impulse response is found by deconvolving the input signal from the recorded signal. The analogue of this process within the waveguide mesh is to excite the mesh at one node, and to record the pressure at some output node at each time step, which is then deconvolved as before. Recording node pressures is simple, and deconvolution is a well-known technique. Injecting a source signal, however, requires careful engineering in order to maintain the physical plausibility and numerical robustness of the model. Source design has two main concerns: the method that is used to add the signal into the mesh, and the signal that is injected.

Input Node Update Method

Firstly, the input signal may be injected at a single node, or distributed across several [50]. It is complicated to calculate an appropriate distributed signal [51], and so single-node sources are more common in the literature [50], [52]–[55].

There are two main options for updating the source node with the input signal. The first method, known as a *hard source*, simply overwrites the pressure value at the source node with that of the input

signal. Hard sources are simple to implement, and ideally couple the input signal to the mesh, but also scatter any incident wave-fronts [56]. Although the initially radiated field is optimal, this spurious scattering is an undesirable artefact. Furthermore, the abrupt pressure discontinuity introduced by the hard source can cause more artefacts through the accumulation of numerical errors. For these reasons, hard sources are generally unsuitable for high-accuracy simulations.

The second method, the *soft source*, instead adds or superimposes the input signal on the standard mesh update equation. Soft sources obey the mesh update equations, and so do not suffer from the scattering problem. However, they do not couple the input signal to the mesh, so that the radiated wave front does not resemble the input signal. For input signals with DC components, soft sources can lead to *solution-growth*, in which the DC level of the entire simulation increases exponentially over time. This is not to do with numerical error or stability. Rather, the DC component suggests that the input signal is of infinite length, and that the source continues to create volume velocity for the duration of the simulation [57]. That is, the model is valid, but it is being used to model a physically-implausible situation. Solution-growth is unacceptable for modelling purposes. In all cases it creates an increasing DC component which is difficult to remove from the output. It also leads to a loss of precision in floating-point simulations, as intervals between floating-point numbers are larger for higher-magnitude numbers. The low-magnitude content of interest cannot be specified with high precision when it is superimposed on a high-magnitude DC component. In the worst case, the DC component will build up so much that the numerical representation "overflows", placing an upper bound on the duration of the simulation. Some of the drawbacks of the soft source can be alleviated by carefully constructing the input signal, which will be discussed later.

Solution growth is not seen in hard source models, because the source node pressure is replaced by that of the input function. Therefore, at the source node, there is no addition of DC level. Unfortunately, rarefaction cannot occur at the source node, which can cause low-frequency oscillations instead. These oscillations are easier to remove than an increasing DC offset, and have less impact on precision.

A special case of the soft source is the *transparent source*, described in [56]. This method is the same as the soft source, but additionally subtracts the convolution of the mesh impulse response with the input signal from the input node. The resulting wave-front has the same characteristics as that produced by a hard source, with the additional benefit that incident wave-fronts are not scattered. The drawback of this technique is that the mesh impulse-response must be precomputed before the simulation can begin, which is achieved by running a separate (costly) waveguide mesh simulation. Once found, the mesh response can be used for any other simulation sharing the same mesh topology. Although the transparent source is complex to set up, its characteristics seem perfect. It behaves as if the input signal is perfectly coupled to the mesh, and there is no scattering problem. However, it retains the solution-growth issue of the general soft source.

The hard and soft source methods can be combined, in order to benefit from the characteristics of both methods. A *time-limited pulse* method is introduced in [50], in which the source node starts as a hard source, and reverts to a soft source after a certain period of time has elapsed. The input signal is ideally coupled to the mesh for its duration, but thereafter the mesh can update as usual. As long as the source is positioned away from reflective boundaries, this solves the scattering issue. However, if the source is near to a reflective boundary, reflected wave-fronts might reach the source node before it has reverted to the standard update equation, scattering them. This input scheme has similar performance to the transparent source, with much reduced complexity, but it also shares a major drawback. If the input signal has a DC component, the time-limited hard source can still cause solution-growth [57]. In addition, if the nodes next to the source are not zero when the update equation switches from hard to soft, the pressure discontinuity introduced may introduce error which will propagate for the remainder of the simulation [53].

On balance, it seems as though the transparent source is the optimum input method. Once the mesh impulse response has been computed, the update method itself is simple to implement, and does not carry any performance penalties. The only disadvantage is that the input function must be carefully designed to have no DC component, to avoid solution-growth. Unfortunately, overcoming this weakness

is somewhat difficult, as will be shown in the following subsection.

Input Signal

Of course, the input signal must be bounded in time, so that the simulation can end. The shorter the input signal, the fewer simulation steps must be computed, so shorter inputs are more efficient. If a single-sample Dirac-delta is used, all frequencies in the output bandwidth will be excited equally, and the output will not require deconvolution, which is another computational saving. It is plain that a short impulsive input is desirable in terms of efficiency. Unfortunately, impulsive signals have a DC component, which can cause solution-growth with soft, particularly transparent, sources.

The constraints of the input signal are, therefore, as follows: The signal shall have no DC component. It shall be as short as possible, to minimise the simulation time. It shall have a wide and flat passband, to increase the accuracy of the deconvolution. The time-compactness and bandwidth constraints are mutually exclusive, so in practice the input length must be chosen to balance bandwidth and calculation time.

Particular constraints of source signals are presented in greater detail in [53], which puts forward an additional constraint, known as the *differentiation constraint*: The input signal shall be equal to the first time derivative of fluid emergence. Fluid emergence should start and end at zero, which in turn enforces a null DC component.

Some particular possibilities for the input signal are the *sine-modulated Gaussian pulse*, the *differentiated Gaussian pulse*, and the *Ricker wavelet*. All of these signals satisfy the differentiation constraint and the length constraint. However, they all have non-flat pass-bands, as shown in the following figure (**??**). A final option is the *physically constrained source* (PCS) model presented in [53]. This method can be used to create input signals with pass-bands much flatter than those of the more conventional pulse and wavelet signals.

TODO frequency response of inputs

A PCS input signal, combined with the transparent source input method, seems perfect for this application. It obeys the differentiation constraint, has a wide and flat passband, and is short in time. The soft source will not cause scattering artefacts, and as the input signal has no DC component, it should not introduce solution-growth.

A test was devised to ensure that the source injection method did not cause solution-growth. A standard rectilinear waveguide mesh with a sampling frequency of 10KHz was set up within a cuboid room, measuring $5.56 \times 3.97 \times 2.81$ metres. A source was placed at (4.8, 2.18, 2.12), and a receiver at (4.7, 2.08, 2.02). The walls of the room were set to have a uniform broadband absorption of 0.006 (see the Boundary Modelling section). "Transparent" input signals were created from a differentiated Gaussian pulse, a sine-modulated Gaussian pulse, and a Ricker wavelet, all of which were set to a centre frequency of $0.05\ f_s$. A physically-constrained source signal was also generated, using parameters suggested in section V of [53]: a max-flat finite-impulse-response pulse-shaping filter kernel with 16 taps and centre frequency of $0.075\ f_s$ and magnitude 250N was generated; it was passed through a mechanical shaping filter with radius 5cm, mass 25g, lower cutoff 100Hz, and resonance 0.7; then this signal was passed through an infinite-impulse-response *injection filter*, and used as a soft source. (These parameters are reproduced here to ensure that the test is repeatable, but a full discussion of their meaning is beyond the scope of this paper. The interested reader is directed to [53] or to the implementation of the physically-constrained source in the Wayverb repository.) Finally, the simulation was run for around 85000 steps (less than the expected Sabine RT60 of the room) with each of the four sources, and the response at the receiver was recorded.

The results of the experiment are shown in the following figure (**??**). The response of a transparent Dirac source (which has a strong DC component) is also shown. All the sources with no DC component

show significantly less solution-growth than the transparent Dirac source. However, they *do* all show the effects of solution-growth.

TODO solution growth results

The solution-growth seen here only becomes prominent towards the end of the simulation, after around 60000 steps. However, papers which propose countermeasures to the solution-growth problem generally only test their solutions up to 15000 steps or so [50], [53], [57]. The results of testing the solution in [55] are not even presented. It is entirely possible that these input methods have not been tested in such a long simulation before.

The reason for the solution-growth is not clear. In general, the problem is caused by repeated superposition of the DC level, which is reflected from boundaries. The waveguide has no inherent way of removing DC, so *any* small DC component will accumulate over time. If the original signal does not have a DC component, then the DC is being added from elsewhere. The most likely origin is numerical error in the waveguide mesh. The experiment above uses 32-bit single-precision floating-point to represent the pressure of mesh nodes, which is necessary because using double-precision would double the memory usage and halve the computational throughput. It is possible that error in these single-precision calculations manifests as a tiny DC component, which then multiplies as the simulation progresses.

Whatever the reason, it is clear that using a soft source generally causes a DC offset to accumulate, even when the input signal has no DC component. Soft sources may be suitable for shorter simulations, however, without running the simulation, it is impossible to know how much DC will accumulate, and whether or not it will remain within acceptable bounds. Therefore, in general, current forms of the soft source are not appropriate for arbitrary room simulation.

As an alternative to a soft source, Wayverb currently uses a hard source with a Dirac impulse as its input method. Though this still causes low-frequency error [53], this error manifests as an oscillation rather than an exponential growth. The oscillation tends to be below the audible range, and therefore can be removed without affecting the perceived quality of the simulation. This removal is achieved with no phase modifications by transforming the signal into the frequency domain, smoothly attenuating the lowest frequencies, and then converting back to the time domain. The main drawback of the hard source is its scattering characteristic. Though undesirable, this behaviour has a physical analogue: in a physical recording, reflected wave-fronts would be scattered from the speaker cabinet or starter pistol used to excite the space.

Code for modelling soft sources remains in the Wayverb repository, so if future research uncovers a soft source without solution-growth, it will be easy to replace the current source model with an improved one.

Implementation

Here, the final waveguide, as implemented in Wayverb, is described.

TODO mesh setup - nodes inside / outside, storage method TODO sample rate, update equations etc. Courant number

TODO mention boundary setup?

5 Hybrid

Background

Crossover Position

There is no concrete rule to place the crossover between "low" and "high" frequencies in this context. It should be clear that, when the time and computing power is available, the cutoff should be placed as high as possible, so as to use accurate wave-based modelling for the majority of the output. However, in practice, it might be useful to have an estimate for the frequencies where wave-modelling is strictly required, to guide the placement of the cutoff frequency. The *Schroeder frequency* is such an estimate, and is based on the density of room resonances or "modes". Below the Schroeder frequency, room modes are well separated and can be individually excited. Above, the room modes overlap much more, resulting in a more "even" and less resonant sound. The Schroeder frequency is defined as follows (see [30, p. 84] for a detailed derivation):

(23)
$$2000\sqrt{\frac{RT60}{V}}$$

Here, $RT60$ is the time taken for the reverb tail to decay by 60dB, and V is the room volume in cubic metres. Note that the Schroeder frequency is inversely proportional to the square root of the room volume. This implies that in larger rooms, the cutoff between modal and non-modal behaviour is lower, and therefore wave-based methods will be required to compute a smaller portion of the spectrum. In turn, this helps to keep the computational cost of wave-based simulations within reasonable bounds. Although the cost of wave-based methods increases with output frequency and simulation size, required output frequency decreases with simulation size. As a result, a hybrid acoustic simulator should be able to simulate even very large enclosures with reasonable accuracy, without incurring excessive costs.

6 Microphone Modelling

Coming soon.

7 Boundary Modelling

Introduction

The ideal boundary model would allow complete control over the frequency- and direction-dependent absorption and scattering of a surface. Though this is reasonably straightforward in geometric models, it is far from a solved problem for the digital waveguide mesh (DWM). Several possible implementations are discussed in the literature, each with unique drawbacks.

This section will begin by discussing the ideal behaviour of modelled acoustic boundaries. Then, the implementation for geometric models will be discussed. Possibilities for DWM boundary models will be investigated, and the final choice of method explained. The geometric and DWM implementations will be evaluated and compared, to ensure equivalence.

Background

The books by Vorlander [31, p. 35] and Kuttruff [30, p. 35] both devote entire chapters to the topic of sound reflection and scattering. Please refer to these books for a detailed and broad explanation of reflection effects. To avoid unnecessary duplication, this background will be brief, aiming only to put terminology and decisions in context.

Magnitude and Phase

The reflection factor R is a complex value given by

(24)
$$R = |R| \exp(i\chi)$$

which describes a modification to the amplitude and phase of the reflected wave ($|R|$ is the magnitude term, χ is phase).

This factor depends both on the frequency and direction of the incident wave. When $\chi = \pi$, $R = -1$, corresponding to a phase reversal. This is known as a "soft" wall, but is rarely seen in room acoustics. It is reasonable to assume that reflections are in-phase in the majority of architectural acoustics problems.

The wall impedance Z is defined as the ratio of sound pressure to the normal component of particle velocity at the wall surface. It is related to the reflection factor by

(25)
$$R = \frac{Z \cos\theta - Z_0}{Z \cos\theta + Z_0}$$

where θ is the angle of incidence, and Z_0 is the characteristic impedance of the propagation medium, normally air. In the case that the wall impedance is independent of the wave angle-of-incidence, the surface is known as *locally reacting*. A locally reacting surface does not transmit waves tangentially along the wall surface.

The absorption coefficient α of the wall is related to the reflection factor by the equation $\alpha = 1 - |R|^2$. It describes the proportion of incident energy lost during reflection.

Properties of surfaces in an acoustic simulation may be described fully by either the reflection factor or impedance. The absorption coefficient fails to encode the phase-change properties of the surface, and so cannot fully describe the full range of possible characteristics. However, if surfaces are assumed to be "hard", i.e. they do not induce phase changes, then the absorption coefficient is an adequate descriptor.

Scattering

The reflection factor, absorption coefficient, and wall impedance describe the behaviour of perfectly-reflected (specular) waves. If the reflecting surface has imperfections or details of the same order as the wavelength, as many surfaces in the real world do, then some components of the reflected wave will be *scattered* instead of specularly reflected.

Describing the nature of the scattered sound is more complicated than specular reflections. A common method is to use a *scattering coefficient*, s, which describes the proportion of outgoing energy which is scattered, and which may be dependent on frequency (see figure 7.1):

(26)

$$E_{\text{scattered}} = E_{\text{incident}}(1-\alpha)s, E_{\text{specular}} = E_{\text{incident}}(1-\alpha)(1-s), E_{\text{total}} = E_{\text{incident}}(1-\alpha)$$

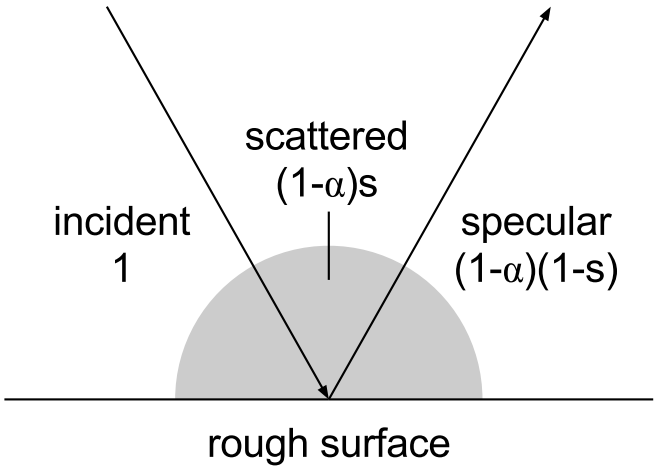

Figure 7.1: Reflected components from a rough surface.

Alone, the scattering coefficient fails to describe the directional distribution of scattered energy. In the case of an ideally-diffusing surface, the scattered energy is distributed according to Lambert's cosine law. That is, the intensity depends only on the cosine of the outgoing scattering angle, and is independent of the angle of incidence (see figure 7.2). More complex scattering distributions, which also depend on the outgoing direction, are possible [41], [58], but there is no single definitive model to describe physically-accurate scattering.

Geometric Implementation

The geometric implementation of reflective surfaces is straightforward. In both image-source and ray tracing methods, each ray starts with a certain intensity or pressure. For specular reflections, the ray pressure must merely be multiplied by the wall reflection coefficient, which may derived from the absorption and scattering coefficients using the equations above. Specifically, the normal-incidence specific impedance ξ_0 is calculated using

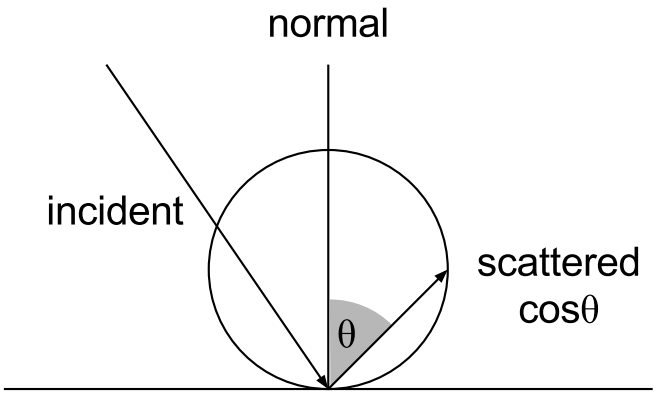

Figure 7.2: Lambert scattering. Scattered intensity is independent of incident angle.

(27)
$$\xi_0 = \frac{1 + R_0}{1 - R_0}$$

where R_0 is the normal-incidence reflection factor. Then, the angle-dependent reflection factor is given by

$$R_\theta = \frac{\xi_0 \cos\theta - 1}{\xi_0 \cos\theta + 1}$$

where θ is the angle of incidence [27].

If the reflection factor is dependent on frequency then the frequency range, and the pressure carried by the ray, must be discretised into bands. Then each band must be modified using a representative reflection factor for that frequency range. This is similar to the approach taken in graphical ray tracing, in which each ray carries separate red, green, and blue components. These components are modified independently, depending on the colour of the reflective surface.

By definition, image-source models find only specular reflections (i.e. image sources), so scattering is not implemented in these models. Scattering can be implemented in ray tracers, but there is no consensus on the optimum method. One option is to spawn two rays at every reflection: a specular ray, and a diffuse ray with random direction. Though this properly replicates the theory, it leads to an explosion in the number of rays which must be traced, so is impractical in most cases. A second option is to decide, using the scattering coefficient as a probability, whether the reflection should be specular or diffuse [2]. This solves the ray-explosion problem, but requires an additional random number to be generated per-reflection, which can be costly for large numbers of rays. An elegant solution is to simply mix the specular and diffuse rays together, using the scattering coefficient as a weighting [59], a technique known as *vector based scattering* [41]. This is the approach taken by Wayverb. A major drawback of all these scattering methods is that the scattering coefficient can only be frequency-dependent if a separate ray is traced for each band. If a single ray is used to carry all frequency components, then each component must be scattered in exactly the same way.

The plain scattering model affects only the ongoing ray direction and amplitude. However, it is worth considering that, at each reflection, the scattered energy may be directly visible to the receiver. This fact is exploited by the *diffuse rain* technique, in which each reflection is considered to spawn a "secondary source" which emits scattered energy towards the receiver. This scattered energy is recorded only if the secondary source is visible from the receiver.

Assuming perfect Lambert diffusion, the magnitude of diffuse rain scattered energy is given by [32, p. 64]:

(28)

$$E_{\text{scattered}} = E_{\text{incident}}(1-\alpha)2s\cos\theta(1-\cos\frac{\gamma}{2})$$

Here, θ is the angle from secondary source to receiver relative against the surface normal, and γ is the opening angle (shown in figure 7.3). The magnitude of the scattered energy depends on the direction from the secondary source to the receiver (by Lambert's cosine law), and also on the solid angle covered by the receiver.

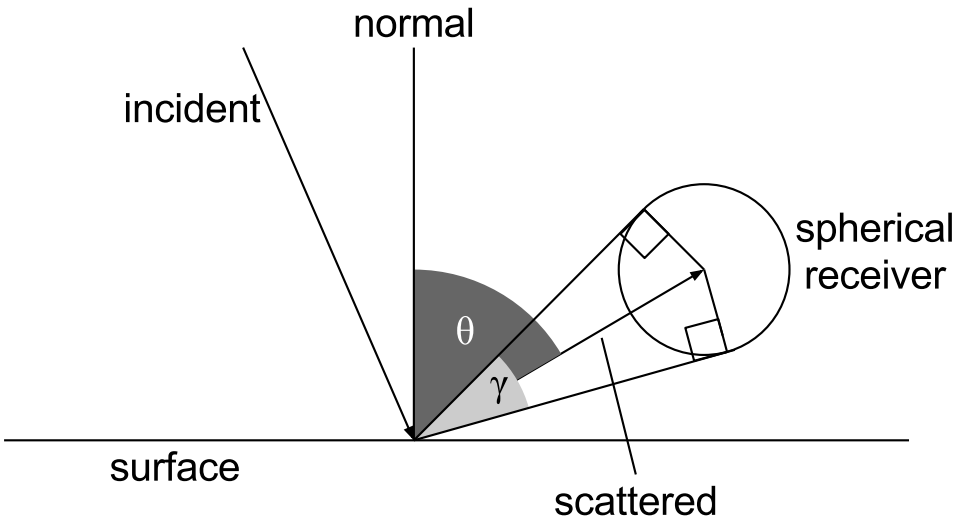

Figure 7.3: Angles used in the diffuse rain equation for a spherical receiver.

DWM Implementation

Modelling boundary surfaces in the digital waveguide mesh (DWM) is a problem that is yet to be solved in a way that is flexible, accurate, and realistic. Methods from the literature each have unique drawbacks, meaning none are particularly satisfactory for applications where realism is required. The two most common methods will be reviewed, and the choice of method for Wayverb will be explained.

Possible Methods

KW-Pipe Technique

This method is described in [60] and [61].

There are two main technically-equivalent formulations of digital waveguides meshes, known as *W-models* and *K-models*. W-models are based on travelling wave variables, which allow for straightforward interaction with a variety of termination types, such as rimguides, fractional delays, or wave digital filters. Wave digital filters, in particular, could be used to model frequency-dependent boundaries and air absorption. However, W-models have great memory requirements, making them impractical for large multi-dimensional simulations. K-models are based on Kirchhoff variables, and depend on physical quantities rather than travelling wave components. Under certain conditions, K-models and finite-difference time-domain (FDTD) simulations are equivalent. FDTD models have much smaller memory requirements than W-models, at the cost of decreased flexibility of filtering, as these models cannot directly interact with wave digital filters.

The KW-pipe is a "converter" between wave- and Kirchhoff- variables, which is designed to allow the majority of a model (that is, the air-filled space inside it) to be constructed as a K-model waveguide mesh. At the boundaries of the model, the KW-pipe is used to connect K-model nodes to W-model nodes. These W-model nodes can then be connected to wave digital filters to simulate frequency-dependent absorption of wave energy. The complete model retains both the memory-efficiency of the K-model and the termination flexibility of the W-model, with the drawback of additional implementation complexity at the interface between the two model types.

This sounds extremely promising, but has a major drawback, as described in [62]: while the inside of the mesh will be 2- or 3-dimensional, the boundary termination afforded by the wave-variable boundary is 1-dimensional. Each boundary node connects to just the closest interior node. As a result, the edges and corners are not considered to be part of the model, as these nodes do not have a directly adjacent interior node. Additionally, the 1D boundary termination equation implies a smaller inter-nodal distance than that of the 2D or 3D mesh interior. This means that when updating an interior node next to a boundary, the inter-nodal distance is greater than when updating the boundary node itself. For these reasons, the 1D termination is unphysical and can lead to large errors in the phase and amplitude of reflections.

Locally Reactive Surfaces Technique

This method, described in [62], aims to create physically correct higher-dimensional boundaries by combining a boundary condition, defined by a boundary impedance, with the multidimensional wave equation. This leads to a model for a *locally reacting surface* (LRS), in which boundary impedance is represented by an infinite-impulse-response (IIR) filter.

As noted above, a surface is locally reacting if the normal component of the particle velocity on the boundary surface is dependent solely upon the sound pressure in front of the boundary. In most physical surfaces, the velocity at the surface boundary will also be influenced by the velocity of adjacent elements on the boundary, so LRS is not a realistic physical model in the vast majority of cases.

However, despite that it is not a realistic physical model, the implementation of the LRS modelling technique is both stable and accurate, as opposed to the 1D KW-pipe termination, which does not accurately model even locally-reacting surfaces.

The LRS model leads to an implementation that is efficient (as it is based completely on the K-model/FDTD formulation) and tunable (boundaries are defined by arbitrary IIR filters).

Choice of Boundary Technique for the DWM

The LRS technique was chosen, as it represented the best compromise between memory efficiency, customization and tuning, and realism. The particular strengths of this model are its performance and tunability, though as mentioned previously it is not physically accurate in many cases. That being said, neither of the boundary models considered are particularly realistic, so even for applications where realism is the most important consideration, the LRS model seems to be the most appropriate.

LRS Implementation

See [62] and [63] for a more detailed explanation.

In the Background section it was noted that the reflection characteristics of a surface are fully defined by a reflection coefficient or wall impedance, both of which might be frequency- and direction-dependent. The LRS technique starts from the definition of wall reflectance:

(29)

$$R = \frac{Z \cos\theta - Z_0}{Z \cos\theta + Z_0}$$

This may instead be written in terms of *specific acoustic impedance* ξ, which is equal to the wall impedance Z divided by the acoustic impedance of the propagation medium (air) Z_0: $\xi = \frac{Z}{Z_0}$.

(30)

$$R = \frac{\xi \cos\theta - 1}{\xi \cos\theta + 1}$$

This equation can be rearranged to define the wall impedance in terms of the reflection coefficient. Further, the reflection coefficient can be replaced with a digital filter $R_0(z)$, describing the frequency-dependent normal-incidence behaviour of the surface. This filter might be derived from per-band absorption coefficients, using the relationship $|R| = \sqrt{1-\alpha}$. In Wayverb, reflection magnitudes are found in this way, and then the Yule-Walker method is used to approximate coefficients for R_0. The substitution leads to an equation for a filter, describing the specific acoustic impedance of the surface:

(31)

$$\xi(z) = \frac{1 + R_0(z)}{1 - R_0(z)}$$

This impedance filter is most efficiently implemented as an *infinte impulse response* (IIR) filter, though surfaces with detailed frequency responses will require high-order filters, which tend to become numerically unstable. The usual solution to this problem would be to split the high-order filter into a series-combination of lower-order filters, however the LRS requires access to intermediate values from the filter delay-line which makes this approach impossible. An alternative solution is suggested in [64], which suggests running the entire simulation multiple times, once for each octave band. This means that the boundary filters can be single-order, and resistant to accumulated numerical error. Compared to high-order boundary filters, this method gives much improved accuracy, but at the (immense) cost of running the entire simulation multiple times. In Wayverb, both approaches are taken, allowing the user to choose between a fast, inaccurate single-run simulation with high-order filters; or a slow, accurate multi-run simulation with low-order filters.

The implementation of the boundaries themselves in the DWM is fairly simple. Appropriate impedance filter coefficients can be inserted into special update equations, which are found by combining the discrete 3D wave equation with the discrete LRS boundary condition. Recall that the DWM operates by updating each mesh node individually, depending on the states of the immediately adjacent nodes. To model boundaries, the update equation for each of the boundary nodes is replaced.

In the case of a flat wall, the boundary node is adjacent to a single inner-node, and a "1D" update equation is used. Where two perpendicular walls meet, the nodes along the edge will each be adjacent to two "1D" nodes, and a "2D" update equation is used for these nodes. Where three walls meet, the corner node will be directly adjacent to three "2D" nodes, and a "3D" update equation is used for this node. The three types of boundary nodes are shown in the following diagram (7.4). Note that this method is only capable of modelling mesh-aligned surfaces. Other sloping or curved surfaces must be approximated as a group of narrow mesh-aligned surfaces separated by "steps". For example, a wall tilted at 45 degrees to the mesh axes will be approximated as a staircase-like series of "2D" edge nodes.

Test Procedure

Only the LRS method is tested here. The implementation of frequency-dependent boundaries in geometric simulations amounts to multiplication by a reflection coefficient (with optional scattering) per-band, so there is little to test. The LRS waveguide boundary is more complicated, as it embeds IIR filters into the waveguide boundaries, so it is worth testing that the boundary nodes behave as expected.

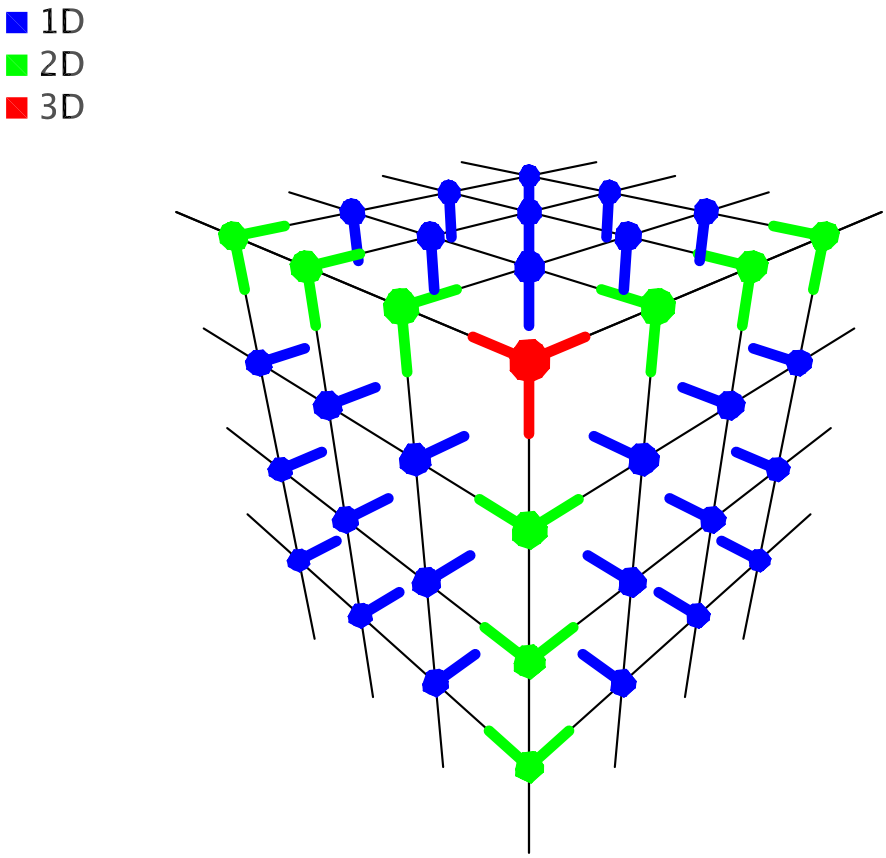

Figure 7.4: The three types of boundary nodes, used to model reflective planes, edges, and corners. 1D nodes are adjacent to inner nodes, 2D nodes are adjacent to two 1D nodes, and 3D nodes are adjacent to three 2D nodes.

The testing procedure used is similar to that in [62]. Code for the test can be seen at https://github.com/reuk/wayverb/blob/master/bin/boundary_test/boundary_test.cpp.

Simulation Parameters

- Cubic room, $300 \times 300 \times 300$ nodes.
- 8 KHz mesh sampling frequency.
- Run for 420 steps.
- Source and receiver placed 37 node-spacings from the centre of the boundary.

The following diagram (7.5) shows the testing setup, which will be explained in the next section.

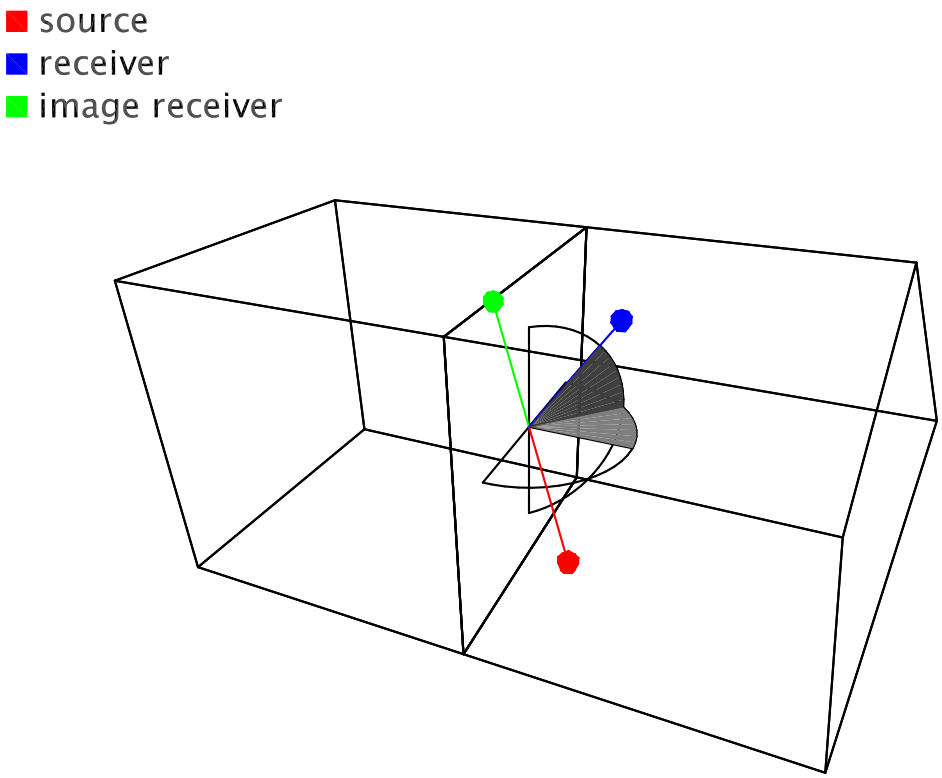

Figure 7.5: The setup of the two room-sizes, and the positions of sources and receivers inside.

Method

A simulation was run using the parameters above, and the reflected signal at the receiver was recorded. This first recording, r_f, contained a direct and a reflected response. Then, the room was doubled in size along the plane of the wall being tested, essentially removing this boundary from the model. The simulation was run again, recording just the direct response at the receiver (r_d). Finally, the receiver position was moved to its reflected position "through" the tested wall, and the simulation was run once more, producing a free-field response (r_i).

The reflected response was isolated by subtracting r_d from r_f, cancelling out the direct response. This isolated reflection is r_r. To find the effect of the frequency-dependent boundary, the frequency content

of the reflected response was compared to the free-field response r_i. This was achieved by windowing r_r and r_i with the right half of a Hanning window, then taking FFTs of each. The experimentally determined numerical reflectance was determined by dividing the absolute values of the two FFTs.

To find the accuracy of the boundary model, the numerical reflectance was compared to the theoretical reflection of the digital impedance filter being tested, which is defined as:

(32)
$$R_{\theta,\phi}(z) = \frac{\xi(z)\cos\theta\cos\phi - 1}{\xi(z)\cos\theta\cos\phi + 1}$$

where θ and ϕ are the reflection azimuth and elevation respectively.

The test was run for three different angles of incidence, with azimuth and elevation of 0, 30, and 60 degrees respectively. Three different sets of surface absorption coefficients were used, giving a total of nine combinations of source position and absorption coefficients. The specific absorption coefficients are those suggested in [64], shown in the following table:

band centre frequency / Hz	31	73	173	411	974
plaster	0.08	0.08	0.2	0.5	0.4
wood	0.15	0.15	0.11	0.1	0.07
concrete	0.02	0.02	0.03	0.03	0.03

The boundary filter for each material was generated by converting the absorption coefficients to per-band reflectance coefficients using the relationship $R = \sqrt{1-\alpha}$. Then, the Yule-Walker method from the ITPP library [65] was used to calculate coefficients for a sixth-order IIR filter which approximated the per-band reflectance. This filter was converted to an impedance filter by $\xi(z) = \frac{1+R_0(z)}{1-R_0(z)}$, which was then used in the boundary update equations for the DWM.

Results

The results are shown in the following figure (7.6). Although the waveguide mesh has a theoretical upper frequency limit of 0.25 of the mesh sampling rate, the 3D FDTD scheme has a cutoff frequency of 0.196 of the mesh sampling rate for axial directions. This point has been marked as a vertical line on the result graphs.

Evaluation

The initial impression of the results is that the measured response is reasonably accurate, to within 6dB of the predicted response, but only below 0.15 of the mesh sampling rate. Above this point, the responses become very erratic, with very large peaks and troughs. This is especially true for the on-axis (0-degree) tests, where some erratic behaviour is seen as low as 0.12 of the mesh sampling rate. This may be due to numerical dispersion in the waveguide mesh, which is greatest along axial directions [62]. At the other end of the spectrum, the results look acceptable, adhering closely to the predicted response.

The poor performance at the top of the spectrum is not particularly concerning, as the waveguide mesh is designed to generate low-frequency content. If wideband results are required, then the mesh can simply be oversampled. To prevent boundary modelling error affecting the results of impulse response synthesis in the Wayverb app, the mesh cutoff frequency is locked to a maximum of 0.15 of the mesh sampling rate.

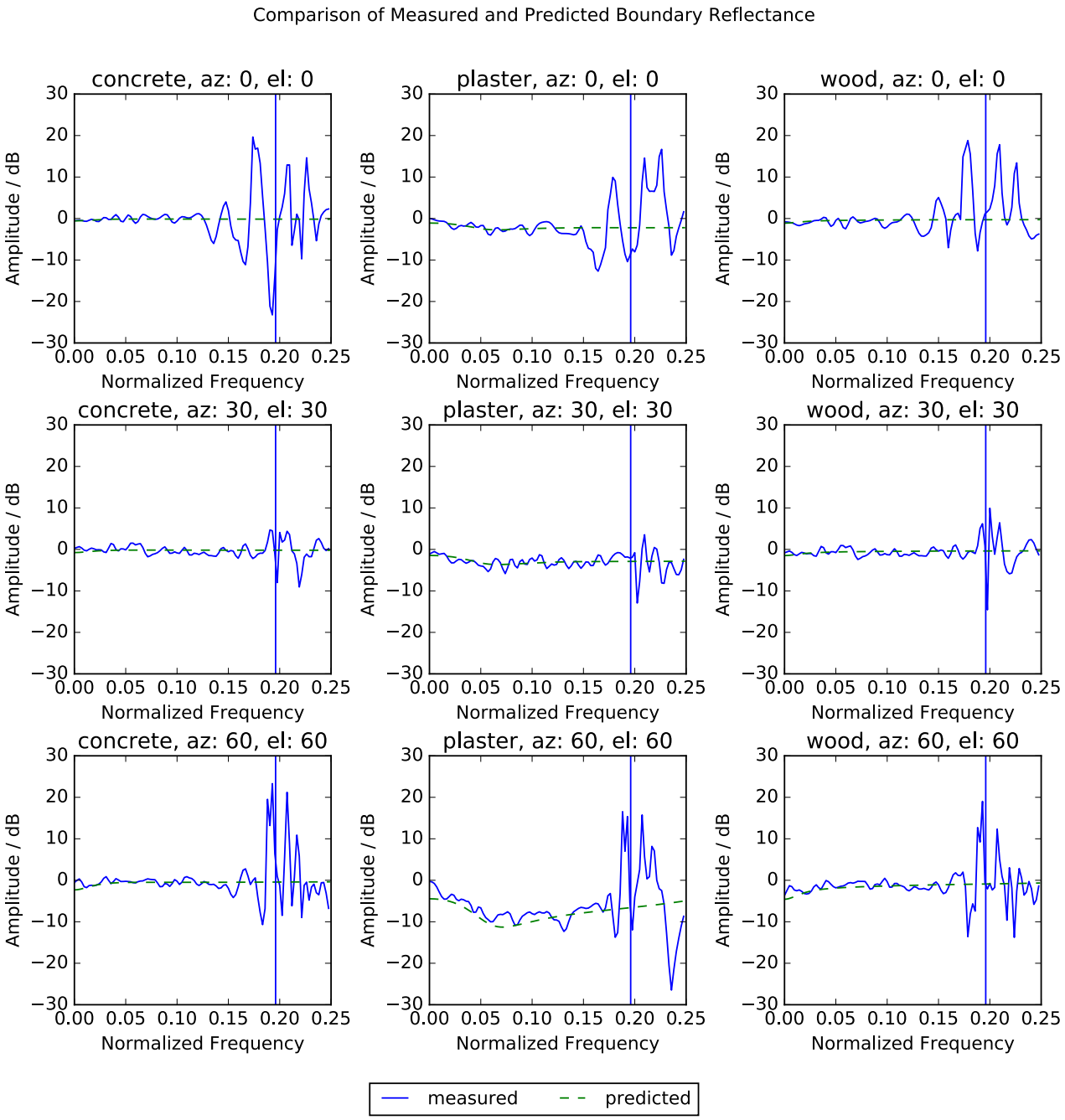

Figure 7.6: Measured boundary reflectance is compared against the predicted reflectance, for three different materials and three different angles of incidence.

It is also worth noting that ideally this experiment would be conducted with a completely flat wave-front, which is not easily accomplished. The experiments in [62] use large meshes (around 3000 by 3000 nodes, nine million in total) and place the sources a great distance away from the boundary being studied in order to maintain a mostly-flat wave-front. However, the experiments are only run in two dimensions. In fact, no experimental results are given for the implementation of three-dimensional boundaries. This is probably because running a 3D simulation on a similar scale would require a mesh of twenty-seven billion nodes, which in turn would require gigabytes of memory and hours of simulation time.

According to [62], in some of the experiments with 2D meshes, there are disparities at low frequencies between the predicted and actual results, which they say is an artefact of non-flat wave-fronts. Interestingly, there is little low-frequency error in the experimental results above, despite the fact that the source is placed very close to the boundary, and the wave-front is therefore very rounded. This might, however, be the cause of the relatively small broadband fluctuations between 0 and 0.15 of the mesh sampling rate. The filters used in this test are also of much higher order than those tested in [62], giving a greater chance of accumulated numerical error. This may be the cause of the volatile high-frequency behaviour.

In conclusion, for the most part, the results presented adhere closely to the expected results, with the caveat that the surface reflectance is only accurate at low frequencies, below around 0.15 of the mesh sampling rate. Different absorption coefficients lead to clearly-different reflectance coefficients, which are additionally accurate at multiple angles of incidence. While not completely accurate, this model is both fast and tunable, making it a good candidate for boundary modelling in room acoustics simulations.

8 Evaluation and Future

Coming soon.

References

This list contains all items cited in the text. It also contains some items not directly mentioned, but which nonetheless guided the development of the project, or might shape its future.

---

[1] A. Southern, S. Siltanen, and L. Savioja, "Spatial room impulse responses with a hybrid modeling method," in *Audio Engineering Society Convention 130*, 2011.

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

[3] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[4] P. Svensson and U. R. Kristiansen, "Computational modelling and simulation of acoutic spaces," in *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*, 2002.

[5] S. B. Shelley, *Diffuse boundary modelling in the digital waveguide mesh*. University of York, 2007.

[6] D. T. Murphy and D. M. Howard, "Digital waveguide mesh topologies in room acoustics modelling," PhD thesis, Citeseer, 2000.

[7] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[8] G. R. Campos and D. M. Howard, "On the computational efficiency of different waveguide mesh topologies for room acoustic simulation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1063–1072, 2005.

[9] L. Savioja and V. Valimaki, "Reduction of the dispersion error in the interpolated digital waveguide mesh using frequency warping," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, 1999, vol. 2, pp. 973–976.

[10] S. A. Van Duyne and J. O. Smith, "The tetrahedral digital waveguide mesh," in *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, 1995, pp. 234–237.

[11] L. Savioja and V. Valimaki, "Interpolated 3-D digital waveguide mesh with frequency warping," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE*

International Conference on, 2001, vol. 5, pp. 3345–3348.

[12] "Odeon." 2016.

[13] "CATT-Acoustic." 2016.

[14] "OTL." 2016.

[15] "EASE." 2016.

[16] "Audioborn." 2016.

[17] D. Schröder and M. Vorländer, "RAVEN: A real-time framework for the auralization of interactive virtual environments," in *Forum Acusticum*, 2011.

[18] M. J. Beeson and D. T. Murphy, "RoomWeaver: A digital waveguide mesh based room acoustics research tool," in *Proc. COST G6 Conf. Digital Audio Effects (Naples, Italy, October 2004)*, 2004, pp. 268–73.

[19] "Ear," *GitHub*. 2016.

[20] "Pachyderm Acoustic," *GitHub*. 2016.

[21] "ParallelFDTD," *GitHub*. 2016.

[22] "I-Simpa," *I-Simpa*. 2016.

[23] "Odeon FAQ." 2016.

[24] A. Southern and S. Siltanen, "A hybrid acoustic model for room impulse response synthesis," in *Proceedings of Meetings on Acoustics*, 2013, vol. 19, p. 015113.

[25] M. Aretz, R. Nöthen, M. Vorländer, and D. Schröder, "Combined broadband impulse responses using FEM and hybrid ray-based methods," in *EAA Symposium on Auralization*, 2009.

[26] D. Murphy, M. Beeson, S. Shelley, A. Moore, and A. Southern, "Hybrid room impulse response synthesis in digital waveguide mesh based room acoustics simulation," in *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, 2008, pp. 129–136.

[27] A. Southern, S. Siltanen, D. T. Murphy, and L. Savioja, "Room impulse response synthesis and validation using a hybrid acoustic model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1940–1952, 2013.

[28] M. Vorlander, "Simulation and auralization of broadband room impulse responses," in *Tecniacústica 2009*, 2009.

[29] A. Krokstad, S. Strom, and S. 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.

[30] H. Kuttruff, *Room Acoustics, Fifth Edition*. CRC Press, 2009.

[31] M. Vorländer, *Auralization: Fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*. Springer Science & Business Media, 2007.

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

[33] A. Alpkocak and M. Sis, "Computing impulse response of room acoustics using the ray-tracing method in time domain," *Archives of Acoustics*, vol. 35, no. 4, pp. 505–519, 2010.

[34] S. A. Van Duyne and J. O. Smith III, "The 3D tetrahedral digital waveguide mesh with musical applications," in *Proceedings of the 1996 International Computer Music Conference*, 1996, pp. 9–16.

[35] L. Savioja, T. Lokki, and V. Välimäki, "The interpolated 3-D digital waveguide mesh method for room acoustic simulation and auralization," *Ultragarsas" Ultrasound"*, vol. 48, no. 3, pp. 48–52, 2014.

[36] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2011.

[37] "Visual Studio support for C++ language features." 2016.

[38] "Clang support for C++ language features." 2016.

[39] "Download LLVM releases." 2016.

[40] J. Amanatides, A. Woo, and others, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, 1987, vol. 87, pp. 3–10.

[41] C. L. Christensen and J. H. Rindel, "A new scattering method that combines roughness and diffraction effects," in *Forum Acousticum, Budapest, Hungary*, 2005.

[42] Z.-h. Fu and J.-w. Li, "GPU-based image method for room impulse response calculation," *Multimedia Tools and Applications*, pp. 1–17, 2016.

[43] R. Heinz, "Binaural room simulation based on an image source model with addition of statistical methods to include the diffuse sound scattering of walls and to predict the reverberant tail," *Applied Acoustics*, vol. 38, no. 2, pp. 145–159, 1993.

[44] J. Antoni, "Orthogonal-like fractional-octave-band filters," *The Journal of the Acoustical Society of America*, vol. 127, no. 2, pp. 884–895, 2010.

[45] J. O. Smith III, "On the equivalence of the digital waveguide and finite difference time domain schemes," *arXiv preprint physics/0407032*, 2004.

[46] M. Karjalainen and C. Erkut, "Digital waveguides versus finite difference structures: Equivalence and mixed modeling," *EURASIP Journal on Applied Signal Processing*, vol. 2004, pp. 978–989,

2004.

[47] J. Botts and L. Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 171–175.

[48] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic modeling using the digital waveguide mesh," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 55–66, 2007.

[49] J. Sheaffer and B. M. Fazenda, "FDTD/K-DWM simulation of 3D room acoustics on general purpose graphics hardware using compute unified device architecture (CUDA)," *Proc. Institute of Acoustics*, vol. 32, no. 5, 2010.

[50] H. Jeong and Y. W. Lam, "Source implementation to eliminate low-frequency artifacts in finite difference time domain room acoustic simulation," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 258–268, 2012.

[51] S. Sakamoto, "Phase-error analysis of high-order finite difference time domain scheme and its influence on calculation results of impulse response in closed sound field," *Acoustical Science and Technology*, vol. 28, no. 5, pp. 295–309, 2007.

[52] Y. W. Lam and J. A. Hargreaves, "Time domain modelling of room acoustics," in *Proceedings of the Institute of Acoustics*, 2012.

[53] J. Sheaffer, M. van Walstijn, and B. Fazenda, "Physical and numerical constraints in source modeling for finite difference simulation of room acousticsa)," *The Journal of the Acoustical Society of America*, vol. 135, no. 1, pp. 251–261, 2014.

[54] D. T. Murphy, A. Southern, and L. Savioja, "Source excitation strategies for obtaining impulse responses in finite difference time domain room acoustics simulation," *Applied Acoustics*, vol. 82, pp. 6–14, 2014.

[55] B. Dimitrijevic, B. Nikolic, S. Aleksic, and N. Raicevic, "Optimization of Excitation in FDTD Method and Corresponding Source Modeling," *RADIOENGINEERING*, vol. 24, no. 1, p. 11, 2015.

[56] J. B. Schneider, C. L. Wagner, and S. L. Broschat, "Implementation of transparent sources embedded in acoustic finite-difference time-domain grids," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 136–142, 1998.

[57] J. Sheaffer, M. V. Walstijn, and B. M. Fazenda, "A physically-constrained source model for FDTD acoustic simulation," in *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, 2012.

[58] J. Durany, T. Mateos, and A. Garriga, "Analytical Computation of Acoustic Bidirectional Reflectance Distribution Functions," *Open Journal of Acoustics*, vol. 5, no. 04, p. 207, 2015.

[59] J. H. Rindel, "The use of computer modeling in room acoustics," *Journal of vibroengineering*, vol. 3, no. 4, pp. 41–72, 2000.

[60] D. T. Murphy and M. Beeson, "The KW-boundary hybrid digital waveguide mesh for room acoustics applications," *IEEE Transactions on Audio, Speech, and Language Processing*, vol.

15, no. 2, pp. 552–564, 2007.

[61] A. Kelloniemi, "Frequency-dependent boundary condition for the 3-D digital waveguide mesh," in *Proc. Int. Conf. Digital Audio Effects (DAFx'06)*, 2006, pp. 161–164.

[62] K. Kowalczyk and M. van Walstijn, "Modeling frequency-dependent boundaries as digital impedance filters in FDTD and K-DWM room acoustics simulations," *Journal of the Audio Engineering Society*, vol. 56, no. 7/8, pp. 569–583, 2008.

[63] K. Kowalczyk and M. van Walstijn, "Modelling Frequency-Dependent Boundaries as Digital Impedance Filters in FDTD Room Acoustic Simulations," in *Audio Engineering Society Convention 124*, 2008.

[64] S. Oxnard, D. O'Brien, J. van Mourik, and D. Murphy, "Frequency-Dependent Absorbing Boundary Implementations in 3D Finite Difference Time Domain Room Acoustics Simulations," 2015.

[65] "ITPP homepage." 2013.

[66] J.-F. Allard and Y. Champoux, "New empirical equations for sound propagation in rigid frame fibrous materials," *The Journal of the Acoustical Society of America*, vol. 91, no. 6, pp. 3346–3353, 1992.

[67] J. B. Allen and D. 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.

[68] H. E. Bass, H.-J. Bauer, and L. B. Evans, "Atmospheric absorption of sound: Analytical expressions," *The Journal of the Acoustical Society of America*, vol. 52, no. 3B, pp. 821–825, 1972.

[69] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, and H. Hagen, "Phonon tracing for auralization and visualization of sound," in *VIS 05. IEEE Visualization, 2005.*, 2005, pp. 151–158.

[70] S. Bilbao, "Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1524–1533, 2013.

[71] J. Borish, "Extension of the image model to arbitrary polyhedra," *The Journal of the Acoustical Society of America*, vol. 75, no. 6, pp. 1827–1836, 1984.

[72] G. Campos and D. Howard, "A parallel 3D digital waveguide mesh model with tetrahedral topology for room acoustic simulation," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx),(Verona, Italy)*, 2000, pp. 73–78.

[73] E. Deines, "Comparative visualization for wave-based and geometric acoustics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1173–1180, 2006.

[74] M. E. Delany and E. N. Bazley, "Acoustical properties of fibrous absorbent materials," *Applied acoustics*, vol. 3, no. 2, pp. 105–116, 1970.

[75] F. Fontana and D. Rocchesso, "Signal-theoretic characterization of waveguide mesh geometries for models of two-dimensional wave propagation in elastic media," *IEEE Transactions on Speech*

and Audio Processing, vol. 9, no. 2, pp. 152–161, 2001.

[76] J. Escolano-Carrasco and F. Jacobsen, "A physical interpretation of frequency dependent boundary conditions in a digital waveguide mesh," in *Proceedings of Thirteenth International Congress on Sound and Vibration*, 2006.

[77] C. M. Furse, D. H. Roper, D. N. Buechler, D. A. Christensen, and C. H. Durney, "The problem and treatment of DC offsets in FDTD simulations," *IEEE Transactions on Antennas and Propagation*, vol. 48, no. 8, pp. 1198–1201, 2000.

[78] S. D. Gedney, "Introduction to the finite-difference time-domain (FDTD) method for electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 6, no. 1, pp. 1–250, 2011.

[79] M. Gorzel, G. Kearney, F. Boland, and H. Rice, "Virtual acoustic recording: An interactive approach," in *Proc. of the 13th internation conference on Digital Audio Effects (DAFX). Graz, Austria*, 2010.

[80] H. Hacihabiboglu, B. Gunel, and A. M. Kondoz, "Time-domain simulation of directive sources in 3-D digital waveguide mesh-based acoustical models," *IEEE transactions on audio, speech, and language processing*, vol. 16, no. 5, pp. 934–946, 2008.

[81] H. Hacihabiboglu, B. Gunel, and A. M. Kondoz, "On the accuracy of first-order numerical derivatives in multidimensional digital waveguide mesh topologies," *IEEE Signal Processing Letters*, vol. 15, pp. 9–12, 2008.

[82] H. Hacıhabiboglu, B. Günel, and Z. Cvetkovic, "Simulation of directional microphones in digital waveguide mesh-based models of room acoustics," *IEEE Trans. on Audio, Speech and Language Process*, vol. 18, no. 2, pp. 213–223, 2010.

[83] M. F. Hadi and N. B. Almutairi, "Discrete finite-difference time domain impulse response filters for transparent field source implementations," *IET microwaves, antennas & propagation*, vol. 4, no. 3, pp. 381–389, 2010.

[84] B. Hamilton, "Sampling and Reconstruction on a Diamond Grid and the Tetrahedral Digital Waveguide Mesh," *IEEE Signal Processing Letters*, vol. 10, no. 20, pp. 925–928, 2013.

[85] J. Huopaniemi, L. Savioja, and M. Karjalainen, "Modeling of reflections and air absorption in acoustical spaces: A digital filter design approach," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1997, pp. 19–22.

[86] M. Kim and G. P. Scavone, "Domain decomposition method for the digital waveguide mesh," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009, pp. 21–24.

[87] G. I. Koutsouris, J. Brunskog, C.-H. Jeong, and F. Jacobsen, "Combination of acoustical radiosity and the image source method," *The Journal of the Acoustical Society of America*, vol. 133, no. 6, pp. 3963–3974, 2013.

[88] E. P. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg, "Non-linear approximation of reflectance functions," in *Proceedings of the 24th annual conference on Computer graphics and*

interactive techniques, 1997, pp. 117–126.

[89] E. A. Lehmann and A. M. Johansson, "Prediction of energy decay in room impulse responses simulated with an image-source model," *The Journal of the Acoustical Society of America*, vol. 124, no. 1, pp. 269–277, 2008.

[90] E. C. Levy, "Complex-curve fitting," *IRE transactions on automatic control*, no. 1, pp. 37–43, 1959.

[91] G. Marbjerg, J. Brunskog, C.-H. Jeong, and E. Nilsson, "Development and validation of a combined phased acoustical radiosity and image source model for predicting sound fields in roomsa)," *The Journal of the Acoustical Society of America*, vol. 138, no. 3, pp. 1457–1468, 2015.

[92] S. McGovern, "The image-source reverberation model in an N-dimensional space," in *Proc. 14th Int. Conf. Digital Audio Effects, Paris, France*, 2011, pp. 11–18.

[93] S. J. Miklavcic and J. Ericsson, *Practical implementation of the 3D tetrahedral TLM method and visualization of room acoustics*. Department of Science; Technology (ITN), Campus Norrköping, Linköping University [Institutionen för teknik och naturvetenskap (ITN), Campus Norrköping, Linköpings universitet], 2004.

[94] A. Mouchtaris, S. S. Narayanan, and C. Kyriakakis, "Effcient multichannel audio resynthesis by subband-based spectral conversion," in *Signal Processing Conference, 2002 11th European*, 2002, pp. 1–4.

[95] D. T. Murphy and J. Mullen, "Digital waveguide mesh modelling of room acoustics: Improved anechoic boundaries," in *Proc. DAFX*, 2002, vol. 2, pp. 163–168.

[96] G. Nelson, L. Pfeifer, and R. Wood, "High-speed octave band digital filtering," *IEEE Transactions on audio and electroacoustics*, vol. 20, no. 1, pp. 58–65, 1972.

[97] M. J. Howarth, "The application of advanced computer models to the prediction of sound in enclosed spaces," PhD thesis, University of Salford, 1998.

[98] J. O'Rourke, "Finding minimal enclosing boxes," *International journal of computer & information sciences*, vol. 14, no. 3, pp. 183–199, 1985.

[99] J. Revelles, C. Urena, and M. Lastra, "An efficient parametric algorithm for octree traversal," 2000.

[100] J. H. Rindel, "Modelling the angle-dependent pressure reflection factor," *Applied Acoustics*, vol. 38, no. 2, pp. 223–234, 1993.

[101] L. Savioja, M. Karjalainen, and T. Takala, "DSP formulation of a finite difference method for room acoustics simulation," in *Proceedings of NORSIG'96 IEEE Nordic Signal Processing Symposium*, 1996, pp. 455–458.

[102] L. Savioja and T. Lokki, "Digital waveguide mesh for room acoustic modeling," *CAMPFIRE*, vol. 6, no. 3, p. 13, 2001.

[103] M. R. Schroeder, "New Method of Measuring Reverberation Time," *The Journal of the Acoustical*

Society of America, vol. 37, no. 3, pp. 409–412, Mar. 1965.

[104] J. Sheaffer, C. Webb, and B. M. Fazenda, "Modelling binaural receivers in finite difference simulation of room acoustics," in *Proceedings of Meetings on Acoustics*, 2013, vol. 19, p. 015098.

[105] J. Sheaffer, M. Van Walstijn, B. Rafaely, and K. Kowalczyk, "Binaural reproduction of finite difference simulations using spherical array processing," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2125–2135, 2015.

[106] J. Sheaffer, "From source to brain: Modelling sound propagation and localisation in rooms," PhD thesis, Citeseer, 2013.

[107] S. Shelley and D. T. Murphy, "Diffusion modelling at the boundary of a digital waveguide mesh," in *Signal Processing Conference, 2005 13th European*, 2005, pp. 1–4.

[108] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West, "A beam tracing approach to acoustic modeling for interactive virtual environments," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 21–32.

[109] S. Siltanen, T. Lokki, S. Kiminki, and L. Savioja, "The room acoustic rendering equation," *The Journal of the Acoustical Society of America*, vol. 122, no. 3, pp. 1624–1635, 2007.

[110] S. Siltanen, T. Lokki, S. Tervo, and L. Savioja, "Modeling incoherent reflections from rough room surfaces with image sources," *The Journal of the Acoustical Society of America*, vol. 131, no. 6, pp. 4606–4614, 2012.

[111] S. Siltanen, A. Southern, and L. Savioja, "Finite-difference time domain method source calibration for hybrid acoustics modeling," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 166–170.

[112] A. Southern and D. T. Murphy, "2nd order spherical harmonic spatial encoding of digital waveguide mesh room acoustic models," in *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx07), Bordeaux, France*, 2007, pp. 101–108.

[113] A. Southern and D. Murphy, "Methods for 2 nd Order Spherical Harmonic Spatial Encoding in Digital Waveguide Mesh Virtual Acoustic Simulations," in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2007, pp. 203–206.

[114] J. E. Summers, K. Takahashi, Y. Shimizu, and T. Yamakawa, "Assessing the accuracy of auralizations computed using a hybrid geometrical-acoustics and wave-acoustics method," *The Journal of the Acoustical Society of America*, vol. 115, no. 5, pp. 2514–2515, 2004.

[115] S. Bilbao, "Grid Functions and Finite Difference Operators in 2D," in *Numerical Sound Synthesis*, John Wiley & Sons, Ltd, 2009, pp. 287–304.

[116] A. Foteinou, J. Van Mourik, S. Oxnard, and D. Murphy, "Hybrid Acoustic Modelling of Historic Spaces Using Blender," 2014.

[117] G. S. Warren and W. R. Scott, "Numerical dispersion in the finite-element method using three-dimensional edge elements," *Microwave and Optical Technology Letters*, vol. 18, no. 6, pp.

423–429, 1998.

[118] C. Zhang and T. Chen, "Efficient feature extraction for 2D/3D objects in mesh representation," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001, vol. 3, pp. 935–938.

[119] S. Siltanen, T. Lokki, and L. Savioja, "Rays or waves? Understanding the strengths and weaknesses of computational room acoustics modeling techniques," in *Proceedings of the International Symposium on Room Acoustics*, 2010.

[120] K. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, May 1966.