Interactive Sound Propagation using Precomputation and Statistical Approximations

Lakulish Antani

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill

2013

Approved by:

Gary Bishop

Dinesh Manocha

Ming C. Lin

Anselmo Lastra

Nikunj Raghuvanshi

ABSTRACT

LAKULISH ANTANI: Interactive Sound Propagation Using Precomputation and Statistical Approximations
(Under the guidance of Dinesh Manocha.)

Acoustic phenomena such as early reflections, diffraction, and reverberation have been shown to improve the user experience in interactive virtual environments and video games. These effects arise due to repeated interactions between sound waves and objects in the environment. In interactive applications, these effects must be simulated within a prescribed time budget. We present two complementary approaches for computing such acoustic effects in real time, with plausible variation in the sound field throughout the scene. The first approach, Precomputed Acoustic Radiance Transfer, precomputes a matrix that accounts for multiple acoustic interactions between *all* scene objects. The matrix is used at run time to provide sound propagation effects that vary smoothly as sources and listeners move. The second approach couples two techniques – Ambient Reverberance, and Aural Proxies – to provide approximate sound propagation effects in real time, based on only the portion of the environment immediately visible to the listener. These approaches lie at different ends of a space of interactive sound propagation techniques for modeling sound propagation effects in interactive applications. The first approach emphasizes accuracy by modeling acoustic interactions between all parts of the scene; the second approach emphasizes efficiency by only taking the local environment of the listener into account. These methods have been used to efficiently generate acoustic walkthroughs of architectural models. They have also been integrated into a modern

game engine, and can enable realistic, interactive sound propagation on commodity desktop PCs.

To my parents, Jayshree and Shailesh Antani.

Acknowledgments

The past five years spent working in the Department of Computer Science at UNC Chapel Hill on the research that ultimately formed this dissertation have been memorable. I would like to thank the many people who played an important role in my journey and professional evolution. First of all, I would like to thank my advisor, Prof. Dinesh Manocha, for his guidance and support, as well as the freedom I was provided throughout the course of this work. I would also like to thank the members of my committee, Prof. Gary Bishop, Prof. Ming C. Lin, Prof. Anselmo Lastra, and Dr. Nikunj Raghuvanshi, for their insightful feedback and discussions on my research work and this dissertation.

I would also like to thank the many talented collaborators I have had the privilege of working with, including Anish Chandak, Micah Taylor, Ravish Mehra, Sean Curtis, Hengchin Yeh, and Zhimin Ren. I would also like to thank Prof. Lauri Savioja and Dr. Tapio Lokki at Aalto University, and Adam Lake at Intel Corporation for their collaboration and insight. I would also like to thank the many anonymous reviewers who have helped me improve the quality of my work.

I would like to thank Valve Corporation for permission to use the Source SDK and Half-Life 2 artwork for our demo scenes. I would also like to thank the Army Research Office, the National Science Foundation, and Intel Corporation for their support. This work was supported in part by ARO contract W911NF-10-1-0506, NSF awards 0917040, 0904990, and 1000579, and RDECOM contract WR91CRB-

08-C-0137.

Finally, I am indebted to my friends and family, especially my parents, Jayshree and Shailesh Antani, not only for putting up with the long hours I spent working, but also for providing me with constant, invaluable support and encouragement, without which this work would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer graphics has made significant progress in the last few decades. Video games, film, and animation have driven the development of improved graphics techniques and hardware, as well as popularized their use, to the point that computer graphics is now a household name. In addition to entertainment applications, computer graphics has helped revolutionize the ways in which we interact with computers (through rich graphical user interfaces) and the ways in which we interpret data (through, for example, scientific or medical visualization). In fact, computer graphics techniques such as graphics processing units (GPUs) have made a profound impact on fields as diverse as oil exploration and finance.

Interactive applications such as video games use high-performance, realistic visual rendering, as well as many advanced techniques for physical simulation and character animation. All of these techniques are used to improve the player's sense of immersion in the virtual environment. In recent years, games have used other modalities such as touch and gesture recognition, to further improve player engagement. Sound is another modality used to improve player immersion. Sound designers spend significant amounts of time generating realistic sound effects and tuning the acoustic effects

of virtual environments. However, this process typically involves manual recording of real-world sounds, or manual tuning of acoustic filters [33]. Acoustic simulation or sound rendering techniques are rarely used in interactive applications due to their compute-intensive nature.

The simulation of *sound propagation*, i.e., how sound waves behave in an environment, can be used to add realistic acoustic effects to interactive applications. To simulate sound propagation, we begin with the position of the sound source and the sound waves it emits. This is combined with a description of the 3D environment to simulate sound waves as they travel through the environment until they reach a listener position. This thesis presents techniques for performing sound propagation in interactive applications, to add acoustic effects in real-time.

1.1 Applications

There are a wide range of applications that can benefit from improved (i.e., more accurate, more efficient, or both) sound propagation simulation. Some of these applications are briefly summarized below, along with the manner in which they use (or may use) sound propagation simulation.

1.1.1 Games

Modern video games use advanced, high-performance rendering techniques for improved lighting, shadows, and surface shading, as well as a range of simulation and animation techniques for rigid bodies, fluids, smoke, and humanoid characters to achieve an increased degree of visual realism. Increasingly, video games are turning to other modalities such as touch-based input (with devices such as Apple's iPad) and gesture recognition (with devices such as Microsoft's Kinect), to further improve

Figure 1.1: Examples of video games with gameplay that stands to benefit from sound propagation effects. **Left**: Bioshock [1], a first-person shooter. Sound propagation can help players locate unseen enemies, even from behind cover. **Center**: Thief 3 [21], a stealth game. Sound propagation can help players track and evade enemies. **Right**: Amnesia [26], a survival horror game. Sound propagation can improve player immersion and heighten the emotional experience.

player immersion.

Another vital means of improving player immersion is through realistic audio rendering (or sound rendering). A wide variety of games benefit from the use of improved acoustic effects. In first-person shooter games such as *Bioshock* [1] (Figure 1.1), enemies (including hard-to-spot snipers) may attack the player from multiple directions, and are often located behind cover. In such situations, directional acoustic cues can often help players locate the enemies more quickly, leading to less frustrating gameplay. In stealth-based games such as *Thief* [21] (Figure 1.1), the player must often evade wandering enemies, or track them without being seen. In such situations, too, acoustic cues (particularly those pertaining to reflected or occluded sound) can help players keep track of the whereabouts of enemies without having to maintain a line of sight. In survival horror games such as *Amnesia* [26] (Figure 1.1), improved acoustic effects can significantly improve immersion, and the level of player engagement.

However, current video games rarely use sound propagation simulation to create immersive acoustic effects, instead relying on manual creation of audio filters for this purpose. Usually, only the directionality of direct sound is modeled, based on the relative positions of the sound source and listener. Artist-specified filters are

used for reflected, occluded, or reverberant sound [33]. The main reason for this is that video games require interactive simulation that is performed in real-time as the player moves around in the game environment. The simulation should also be able to handle large, complex scenes with moving sources and moving listeners. Sound [33] propagation effects must be updated around 10–15 times per second, while taking up a relatively small fraction (typically around 20%) of a typical game's frame budget. Most current algorithms for sound propagation simulation are unable to meet these tight constraints, thereby making them impractical for use in video games.

1.1.2 Virtual Reality

Virtual Reality (VR) systems have been used for a variety of purposes, ranging from training [93], therapy [29], tourism [58, 51], and learning [53]. In such settings, acoustic effects can add a significant degree of environmental context to the virtual environment, and can improve the training or therapy process. For example, when VR simulation is used for treating post-traumatic stress disorder (PTSD) [29], acoustic effects can help recreate a believable war experience in a controlled setting, to help treat soldiers suffering from PTSD.

In VR applications, many of the interactivity constraints of video games apply. However, since VR simulators often have more computational resources available to them, as compared to games, it is often desirable to provide more accurate sound propagation simulation results than would be needed for consumer-oriented video games.

1.1.3 Architectural Acoustics

Architectural acoustics involves the use of acoustic principles and acoustic simulation to improve the acoustic properties of architectural designs and buildings. Judicious use of sound propagation simulation can significantly reduce redesign or reconstruction costs that may be incurred due to poor acoustics. Typically, architectural acoustic simulations are run offline [19], since their key requirement is accuracy of the results. However, there may be scenarios where architects or architectural acoustic consultants need to employ acoustic simulation in an interactive manner. One example is interactive prototyping, where architects wish to estimate the acoustic impact of a design change while they modify the design. Another is architectural walkthroughs, where interactive simulation is used to recreate the acoustics of a space while carrying out a walkthrough of the architectural design, either to better understand the acoustics of the design, or to showcase the design to clients.

1.2 Sound Rendering Pipeline

The simulation of sound can be organized into a rough *sound rendering pipeline*, consisting of three stages related to the simulation of the generation, propagation, and reception of sound waves.

1.2.1 Sound Synthesis

Sound synthesis refers to the physical modeling of the processes resulting in the generation of sound waves. Practically, this involves determining the positions of sound sources and the sound waves emitted by them. These may either be manually specified with pre-recorded sound signals (stored in any standard audio file format,

such as Wave or MP3), or through physical simulation of the vibration of sounding objects. In recent years, there has been much research on generating sound from rigid body collisions [34, 61], friction [64], thin shell vibrations [16], rigid body fracture [95], fluids [54, 94], and cloth [5].

Another important aspect of sound sources is their *directivity*. Different sound sources emit sound waves with different amplitudes and phases in different directions. For example, sound from a megaphone is louder in the direction it is pointing in than in other directions. Many techniques have been developed for representing source directivities, including far field approximations [87] and spherical harmonics [55].

1.2.2 Sound Propagation

Sound propagation refers to the modeling of how sound waves spread through an environment after being emitted by the source. This involves modeling the geometry and material properties of the environment.

The geometry is typically represented in discrete manner using triangle meshes or voxel grids, depending on the simulation algorithm used. Acoustic simulation typically does not require geometry to be represented at the same level of detail as visual rendering, but the simplification of geometric models to the complexities suitable for acoustic simulation remains an open problem [69].

The material properties of the scene are typically specified in terms of absorption and scattering coefficients. These are defined for each octave of frequency. Recently, there has been work on acquiring and incorporating direction-dependent material properties [84], inspired by similar work in visual rendering, but this information is often difficult to acquire from real-world objects.

Given information about sound sources, scene geometry, material properties, and listener properties (described in the next section), a sound propagation algorithm is

used to determine the sound signal received by the listener. There are a wide variety of sound propagation algorithms, ranging from numerical methods (Sections 2.1 and 2.2), to ray-tracing-based methods (Section 2.3), to statistical models (Section 2.7), each with its own advantages and disadvantages. These algorithms are used to compute the *sound field*, i.e., the sound wave amplitude as a function of incidence direction and time (or frequency) at the listener position. Sound propagation involves modeling repeated interactions between sound waves and the environment. The number of interactions is often referred to as the "order" of the interaction. For example, a second-order reflection refers to a sound wave that has undergone two reflections. For the purposes of this thesis, low-order refers to an order of 2–4 or less, and higher-order refers to an order of more than 2–4, unless otherwise specified.

1.2.3 Auralization

In the context of this thesis, *auralization* refers to the process of presenting a simulated sound field to the user over a speaker system. Multiple techniques have been developed for auralization, ranging from amplitude panning and Ambisonics for general multi-channel speaker systems, to binaural rendering for accurate sound field reproduction over headphones (Section 2.5.2).

In most cases, sound propagation effects are computed independently of the sound synthesis process, and represented using an *impulse response* (Section 2.4). This impulse response must be convolved with the source sound signal. Interactive applications perform streaming, real-time convolution with time-varying impulse responses using techniques based on the Short Time Fourier Transform (STFT) (Section 2.5.1).

1.3 Challenges

There are several challenges that must be overcome in order to develop a practical method for simulating sound propagation in interactive applications. These are summarized below:

- **Interactive performance**. Sound propagation effects must be computed on-the-fly in interactive applications, and must rapidly update as the source(s) and/or listener move. Auralization (in particular, convolution) must be performed in real-time at audio rates (typically 44.1 kHz) to avoid undesirable audio artifacts. While sound propagation need not be computed as frequently as visual frame updates, the cost of sound propagation, amortized over multiple frames, should take up a small fraction (5–10%) of a typical frame time budget.

- **Storage requirements**. An increasingly popular approach for interactive sound propagation algorithms is to precompute sound propagation between static portions of the scene. However, for these approaches to support moving sources as well as moving listeners, they often require impractically large amounts of data: in some cases several gigabytes of data even for scenes of moderate size and complexity [63]. For practical use in interactive applications, the size of any precomputed data should be kept as small as possible.

- **Performance-quality trade-off**. Interactive applications must scale across a wide variety of hardware. The relative workloads of different components of an interactive application (e.g., rendering, sound, physics, etc.) also tend to vary with time. In such cases, it is beneficial for the sound propagation algorithm to support a means of automatically scaling the quality of the results so as to

satisfy tight performance requirements.

- **Complex, dynamic, and general scenes**. Typical scenes in interactive applications are complex (containing large numbers of detailed objects), large (often spanning several city blocks or more), and often lack special structure (e.g., do not always contains cell-and-portal structures [72]). They often contain moving sources, listeners, and even moving objects. Any practical sound propagation algorithm must handle such environments at interactive rates.

1.4 Thesis Statement

Precomputed acoustic radiance transfer and geometry-based statistical models offer two alternative approaches for adding higher-order sound propagation effects to interactive environments based on application-specific constraints; the first method provides realistic solutions that account for the geometry of the entire environment, and the second provides coarse approximations based on the local environment of the listener.

1.5 Main Contributions

This thesis presents efficient algorithms for adding realistic sound propagation effects to interactive applications. The algorithms are based on two general approaches, both of which bring powerful ideas from visual rendering to the domain of acoustics. First, we present two algorithms based on *precomputed acoustic radiance transfer*, for precomputing sound propagation effects between static portions of the scene, and using this information to efficiently compute sound propagation effects from a moving source to a moving listener. Further, we present two algorithms based on simplified

acoustic models for sound propagation simulation, while using efficient techniques for plausibly varying the parameters of these acoustic models in response to changes in the position and orientation of the listener with respect to the scene geometry.

1.5.1 Direct-to-Indirect Acoustic Radiance Transfer

We present a new algorithm for modeling diffuse reflections of sound based on the direct-to-indirect transfer approach [31]. Our approach is motivated by recent developments in global illumination based on precomputed light transport algorithms [31, 45]. Specifically, our work is based on direct-to-indirect transfer algorithms for visual rendering, which map direct light incident on the surfaces of a scene to indirect light on the surfaces of the scene after multiple bounces. The main novel aspects of this work include:

- **Precomputed Acoustic Radiance Transfer with Moving Sources and Listeners**. The algorithm computes an acoustic transfer operator in matrix form which is decoupled from both the source and the listener positions, and can efficiently update the acoustic response at the listener whenever the source moves.

- **Efficient Acoustic Radiance Transfer using Singular Value Decomposition**. The algorithm approximates the transfer matrix using the singular value decomposition (SVD) to perform higher-order diffuse reflections. We show that this approximation reduces the memory requirements and increases the performance of our algorithm, compared to using acoustic transfer operators without approximation.

1.5.2 Compact Acoustic Transfer Operators

We present a novel geometric sound propagation algorithm that computes diffuse and specular reflections as well as edge diffraction at near-interactive rates. In order to model higher-order reflections and diffraction, our algorithm precomputes an acoustic transfer operator that models how sound energy propagates between surfaces. We use a scene-dependent Karhunen-Loeve transform (KLT) for compactly representing the transfer operators. At run-time, we use a two-pass method that uses the transfer operator to compute higher-order reflections and diffraction, along with interactive ray tracing to model early reflections and diffraction. Some of the main benefits of our approach include:

- **Compact representation**. Our compression technique, based on KLT, results in low memory overhead for the acoustic transfer operators, resulting in a compression factor of up to two orders of magnitude over time-domain or frequency-domain representations.

- **Run-time control between accuracy and performance**. Our choice of basis for representing the acoustic transfer operator has the additional advantage of allowing control over approximation errors, and thereby trading off accuracy for performance in interactive applications.

- **Moving sources and listeners**. Our precomputed acoustic transfer operator is defined in terms of samples distributed over the surfaces of a static scene. As a result, we can efficiently handle moving sources and listeners.

- **Occlusion of sound by dynamic objects**. Our algorithm can handle (to a limited extent) the effect of introducing a dynamic object on the sound field at the listener due to occlusion of sound emitted from the source by moving

obstacles and the subsequent effect of the occlusion on propagated sound, or due to occlusion of propagated sound by moving obstacles before it reaches the listener.

1.5.3 Ambient Reverberance and Aural Proxies

We present a simple and efficient sound propagation algorithm inspired by work on local illumination models (such as ambient occlusion [96]) and the use of proxy geometry in visual rendering. Our approach generates spatially-varying, direction-dependent reflections and reverberation in large scenes at interactive rates. We perform Monte Carlo integration of local visibility and depth functions for a listener, weighted by spherical harmonics basis functions. Our approach also computes a local geometry proxy which is used to compute 2-4 orders of directionally-dependent early reflections, allowing our technique to plausibly model outdoor scenes as well as indoor scenes. Our approach reduces manual effort involved in tweaking reverberation filter parameters, since it automatically generates spatially-varying reverberation based on the scene geometry. Our approach also enables immersive, direction-dependent reverberation due to the use of spherical harmonics to compactly represent directionally-varying depth functions. It is highly efficient, requiring only 5-10 ms to update the reflection and reverberation filters for scenes with tens of thousands of polygons on a single CPU core. Moreover, it is easy to implement and integrate into an existing game, as shown by our integration with Valve's Source engine. We also evaluate our results by comparison against a reference image source method, and through a preliminary user study.

1.6 Thesis Outline

The rest of this thesis is organized as follows:

- In **Chapter 2**, we review prior art in the fields of sound propagation simulation and auralization, with a particular focus on interactive applications.

- In **Chapter 3**, we introduce *acoustic transfer operators*, which are essentially scene-dependent matrices which encapsulate sound propagation effects. We present their derivation from the acoustic rendering equation, as well as an overview of using transfer operators for adding sound propagation effects.

- In **Chapter 4**, we describe a frequency-domain algorithm for computing and storing acoustic transfer operators that model purely diffuse reflections of sound. We also describe a technique based on the Singular Value Decomposition (SVD), which allows higher-order diffuse reflections to be efficiently added to the transfer operator without performing higher-order ray tracing.

- In **Chapter 5**, we describe a time-domain algorithm for computing and storing acoustic transfer operators that model both diffuse and specular reflections of sound, as well as edge diffraction (with some restrictions). The technique uses the Karhunen-Loeve Transform to obtain a compact representation of the transfer operators. We also describe a two-pass algorithm to model purely specular early reflections in addition to the higher-order reflections modeled by the transfer operator.

- In **Chapter 6**, we describe an efficient algorithm to use local geometry around a moving listener to quickly estimate spatially- and directionally-varying parameters for artificial reverberation in interactive applications.

- In **Chapter 7**, we describe an efficient algorithm to compute a rectangular proxy shape for the local geometry around a moving listener. We also describe a method for estimating average material properties for the proxy shape, as well as a method for using the proxy to efficiently compute higher-order reflections of sound without performing higher-order ray tracing.

- Finally, in **Chapter 8**, we conclude the thesis, summarizing the main results, discussing limitations of the presented techniques, and suggesting promising avenues for future work.

Chapter 2

Background

Sound is a phenomenon caused by the vibrations of air or some other medium, e.g., water. These vibrations are measured in terms of the variation of pressure of the medium over time. Pressure variations behave as *longitudinal waves*, i.e., the air molecules oscillate in the direction in which the wave propagates. Sound waves (like any other waves) are described as a superposition of sinusoidal waves. A sinusoidal wave is described by its *frequency* ν, *wavelength* λ, and *propagation speed* $c = \nu\lambda$.

As sound waves propagate through an environment, they may exhibit multiple kinds of acoustic phenomena [38], including:

- **Reflection** When a sound wave strikes a solid obstacle, it may give rise to a reflected wave, as per the laws of reflection.
- **Absorption** Upon striking a solid obstacle, the wave may be absorbed by the obstacle, resulting in reflected waves of reduced amplitude.
- **Transmission** Upon striking a solid obstacle, the wave may continue propagating through the obstacle (subject to the physical properties of the obstacle).

- **Interference** When two sound waves encounter each other, the resulting pressure variations are described by a superposition of the two sound waves. This may result in a wave of greater or lesser amplitude than the original waves. This phenomenon is referred to as interference.

- **Diffraction** When sound waves encounter obstacles whose size is comparable to their wavelength, they bend around the obstacle. This phenomenon is referred to as diffraction.

- **Scattering** The aggregate behavior of sound waves upon encountering objects or surfaces with fine structure (surface detail of size comparable to the wavelength of the sound waves) is referred to as scattering. It is essentially a combination of reflection, diffraction, and other phenomena, caused by the individual elements of the fine structure of the scatterer.

To simulate sound propagation, we must calculate the acoustic pressure $P(\mathbf{x}, t)$ in an environment as a function of position $\mathbf{x}$ and time t. Note that acoustic pressure in this context refers to the *difference* between the actual pressure at a point and some mean reference pressure (e.g., standard atmospheric pressure).

2.1 The Acoustic Wave Equation

The variation of pressure in a domain D with boundary ∂D is governed by the *acoustic wave equation* [59]:

$$\nabla^2 P - \frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} = F, \tag{2.1}$$

where $P(\mathbf{x}, t)$ is the pressure at any point $\mathbf{x} \in D$ as a function of time t, and $F(\mathbf{x}, t)$ is a forcing function defined by sound sources. Here, the speed of sound, c,

is assumed constant; for large outdoor spaces this restriction must often be lifted to account for temperature gradients and wind effects.

To complete the problem specification, we must specify the behavior of P on the boundary ∂D. This is specified using some form of *boundary condition*, including:

- **Dirichlet** boundary conditions involve specifying the value of P at each point on the boundary.

- **Neumann** boundary conditions involve specifying the *normal derivative* of P at each point on the boundary, i.e., the component of the pressure gradient normal to the boundary surface:

$$\frac{\partial P}{\partial n} = \nabla P \cdot \mathbf{n}. \tag{2.2}$$

- **Impedance** boundary conditions involve specifying the *specific acoustic impedance* $Z_s(\nu)$ at each point of the boundary. $Z_s(\nu)$ is the ratio of the pressure P and the normal velocity $v_n = \mathbf{v} \cdot \mathbf{n}$:

$$Z_s(\nu) = \frac{P}{v_n}. \tag{2.3}$$

  In general, Z_s is a complex-valued quantity, defined for each frequency.

To solve for acoustic pressure in general domains, we must use numerical methods. We now briefly describe some numerical methods that have been used for solving the wave equation. Since they all solve for pressure as a function of time, they are also referred to as *time-domain* methods.

2.1.1 Finite Difference Method

In the *finite difference method* (FDM or FDTD) [13], the spatial and temporal partial derivatives are approximated by finite difference expressions. In other words, $\mathbf{x}$ and t are discretized, and the derivatives of pressure are expressed as linear functions of pressure sampled at these discrete positions and times, e.g.:

$$\frac{\partial^2 P(x,t)}{\partial t^2} = \frac{P(x,t_{i+1}) - 2P(x,t_i) + P(x,t_{i-1})}{\Delta t^2}, \tag{2.4}$$

where $t_i = i\Delta t$. Δt is also referred to as the time-step of the simulation. Similarly, spatial derivatives are sampled at a *grid resolution* of Δx. As per the Nyquist theorem, to simulate sound waves of wavelength λ, we must have $\Delta x \leq \frac{\lambda}{2}$. Moreover, the time-step must satisfy the Courant-Friedrichs-Levy condition, i.e., $\Delta t \leq \frac{\Delta x}{\sqrt{3}c}$. The implications of these conditions is that the computational complexity of FDTD is $O(\nu^4)$, and its storage complexity is $O(\nu^3)$.

There are multiple variations of the finite difference method, depending on which specific finite difference approximations are used. Each has its pros and cons, but the most important limitation is that the grid must be oversampled to reduce numerical errors [77]. Typically, $\Delta x \leq \frac{\lambda}{8}$ or less is needed for acceptable accuracy [48].

2.1.2 Pseudo-Spectral Method

The *pseudo-spectral time domain method* (PSTD) [48] computes the spatial derivatives of pressure using a discrete Fourier transform (DFT) [14]. In other words, the DFT of pressure over the discretized domain is first computed. Next, the Fourier coefficients of the pressure field are advanced by one time-step using an analytical expression. As a result, numerical errors due to time-stepping are avoided, and the

grid does not need to be oversampled. Also, the DFT can be efficiently computed using the Fast Fourier Transform (FFT) algorithm [15]. However, the asymptotic space and time complexity of this approach is the same as that of FDTD.

2.1.3 Adaptive Rectangular Decomposition

Adaptive rectangular decomposition (ARD) is a recently-proposed method [62] for performing time-domain simulations in complex domains. The method is based on the observation that in a rectangular domain, the wave equation has an analytical solution, where the pressure can be described in terms of the coefficients of a discrete cosine transform (DCT) [2]. These DCT coefficients can be time-stepped using an analytical expression.

Therefore, the domain in decomposed into multiple rectangular subdomains using a greedy flood-fill algorithm. In each subdomain, the DCT-based method is used, and pressure is communicated across subdomain boundaries (or *interfaces*) using a finite difference stencil. In other words, this is a domain decomposition method, where the spatio-temporal analytic solution is used within each subdomain, and finite differencing is used to perform coupling between subdomains.

2.2 The Helmholtz Equation

As an alternative to solving the wave equation, we may assume that the pressure field is *time-harmonic*, i.e., $P(x,t) = \Psi(x,\omega)e^{\iota\omega t}$, where $\omega = 2\pi\nu$ is the angular frequency. As per the Fourier theorem, any periodic function P can be represented as a superposition of sinusoidal functions. Therefore, by using a sufficiently long period, we may approximate any pressure field as a superposition of time-harmonic pressure fields Ψ. This reduces the problem to that of solving the *Helmholtz equation* [59]:

$$\nabla^2\Psi + k^2\Psi = \tilde{F}, \tag{2.5}$$

where $k = \frac{2\pi}{\lambda}$ is the wavenumber. There are multiple numerical methods for solving the Helmholtz equation. Since they operate on a Fourier decomposition of the pressue field, they are also called *frequency-domain* methods.

2.2.1 Finite Element Method

In the *finite element method* (FEM) [81], the domain is discretized using an irregular mesh, whose elements may be of any shape. Typically, though, simple shapes, such as tetrahedra in 3D, are used. With each *vertex*, or *node*, of the mesh, we associate a *basis function* ϕ_i. These basis functions may be of any form as long their partial derivatives can be defined. In practice, though, simple linear functions are commonly used.

The pressure field at any point $x \in D$ is defined as a linear combination of the basis functions:

$$\Psi(x) = \sum_i c_i \phi_i(x). \tag{2.6}$$

Combining the above equation with the Helmholtz equation and the boundary conditions yields a system of linear equations. Moreover, since the basis functions are defined to have *compact support*, i.e., ϕ_i non-zero only for the mesh elements adjacent to node i, the linear system can be represented using a sparse matrix. This sparse matrix solve is performed once for each frequency.

The computational and storage complexity of FEM are dominated by the sparse matrix linear solve. The number of elements in the spatial discretization must be $O(\nu^3)$ to satisfy the Nyquist condition. This results in an $n \times n$ sparse matrix, where

$n = O(\nu^3)$. Since the linear solve must be repeated for each frequency, the storage complexity of FEM is $O(\nu^4)$, as is its computational complexity.

One of the main advantages of FEM over, say, FDTD, is that the mesh may be irregular, and hence may better approximate complex boundaries.

2.2.2 Boundary Element Method

The *boundary element method* (BEM) [20] is based on the Helmholtz-Kirchhoff integral theorem, according to which, the pressure at any point in the interior of the domain can be uniquely determined from the values of pressure (or its normal derivative) at each point on the boundary of the domain. Hence, BEM proceeds in two steps.

First, the boundary is discretized using a surface mesh, typically a triangle mesh. With each node of the boundary, we associate a basis function ϕ_i. The pressure at any point $x \in \partial D$ is again defined as a linear combination of the basis functions:

$$\Psi(x) = \sum_i c_i \phi_i(x). \tag{2.7}$$

Combining the above equation with the integral form of the Helmholtz equation and the boundary conditions yields a system of linear equations, which must be represented using a dense matrix, since the equations describe propagation between each pair of surface mesh elements. Solving this system yields the pressure on the boundary.

Finally, the value of pressure at any interior point is computed by evaluating an integral as per the Helmholtz-Kirchhoff integral theorem. Since the domain itself is not discretized in BEM, the numerical errors are significantly reduced.

As with FEM, the computational and storage complexity of BEM are dominated

by the complexity of solving the linear system. The number of elements in the surface mesh must be $O(\nu^2)$ to satisfy the Nyquist condition. This results in an $n \times n$ dense matrix, where $n = O(\nu^2)$. Since the linear solve must be repeated for each frequency, the storage complexity (i.e., memory required while calculating the solution) of BEM is $O(\nu^5)$, and its computational complexity is $O(\nu^7)$ if direct matrix inversion is used. This complexity can be reduced to $O(\nu^4)$ per frequency using iterative Krylov subspace solvers.

This complexity can be further reduced using *fast multipole methods* (FMM) [30]. FMM approximates the interactions between groups of mesh nodes. The main result of this approximation is that the size of the dense matrix is reduced to $k \times n$, where k is a large constant and $n = O(\nu^2)$. Therefore, the storage complexity of FMM-BEM is $O(\nu^3)$, and its computational complexity is $O(\nu^2 \log \nu)$ per frequency.

2.2.3 Equivalent Source Method

The *equivalent source method* (ESM) [55] is closely related to the BEM, in that it also solves the integral form of the Helmholtz equation. However, instead of solving for pressure sampled on the domain boundary, pressure is sampled on an *offset surface* of the boundary. The pressure on the offset surface is then expressed as a weighted sum of elementary point sources ϕ_i, which are Green's functions for the Helmholtz equation. The main advantage of sampling pressure on the offset surface is that fewer basis functions (point sources) are required to express the pressure with the same degree of accuracy.

The basis expansion of the pressure field is then combined with the integral form of the Helmholtz equation and the boundary conditions, resulting in a system of linear equations, which must be represented using a dense matrix. As a result, the asymptotic complexity of ESM is the same as that of BEM, i.e., the storage

complexity of ESM is $O(\nu^5)$, and its computational complexity is $O(\nu^7)$.

Recently, a precomputation-based algorithm has been developed based on ESM, which uses transfer operators similar to those defined in Chapter 3. This approach allows scattering and diffraction from discrete objects to be simulated in real-time [52].

2.3 Geometric Acoustics

The computational and storage complexity of numerical methods for solving the wave equation or Helmholtz equation grows rapidly with increasing frequency, or with increasing domain size (i.e., area, volume). Therefore, these methods are feasible only for low frequency sounds and small-to-medium-sized spaces. To simulate higher frequencies or larger domains, we typically use *geometric acoustics* techniques.

By making a high-frequency assumption, it is possible to model the propagation of sound waves using *rays* of sound emitted from a sound source. This is analogous to geometric optics, where light is assumed to propagate along rays emitted from a light source. First, the acoustic pressure is written as follows:

$$P(x,t) = A(x)e^{\iota\omega(t-W(x)/c_0)}, \tag{2.8}$$

where $W(x)$ is called the *eikonal*, and expresses the variation of phase with position, and c_0 is a reference speed of sound. As the speed of sound may change with position in large domains (due to local variations of density, temperature, etc.), we denote the speed of sound at any point x by $c(x)$. Substituting into the wave equation, and making a high frequency approximation (i.e., $\omega \to 0$) yields the *eikonal equation* [22]:

$$\nabla W \cdot \nabla W = \frac{c_0^2}{c^2(x)}. \tag{2.9}$$

While the eikonal equation may be solved numerically, a simpler approach is often used. This is based on the observation that the local direction of propagation of sound at any point is parallel to $\nabla W(x)$. This allows sound propagation to be modeled using rays propagating along $\nabla W(x)$. In general, the rays travel along curved paths, as determined by the variation of $c(x)$ over the domain. This approach is often used to simulate underwater or atmospheric acoustics, where the domain sizes make numerical methods impractical.

In domains where the speed of sound is constant (e.g., indoor spaces, or small outdoor spaces), rays travel along straight lines. Upon encountering solid obstacles (such as walls), the rays may be reflected, absorbed, transmitted, or scattered. Geometric acoustics algorithms determine how rays propagate through a domain, and use this information to simulate sound propagation.

The most compute-intensive step of a geometric acoustics algorithm is typically the ray tracing step, i.e., given a ray with its origin and direction, determining the first point of intersection between the ray and the domain boundary, which also includes objects in the scene. Modern geometric acoustics techniques can make use of the latest work on high-performance ray tracing to significantly improve their performance. These techniques use *acceleration structures* such as Bounding Volume Hierarchies (BVHs) [44] or kD-trees [90], along with parallel programming techniques [65, 57], to achieve near-interactive performance.

There are many algorithms for using ray tracing to simulate sound propagation. Most of them fall into one of the following categories.

2.3.1 Image Source Method

For a rectangular domain with smooth, perfectly rigid walls, the wave equation can be solved analytically. This solution can be written in multiple ways. The *normal mode expansion* is used in ARD, to reduce dispersion errors in large spaces. An alternative representation of the solution is the *image source* expansion [4]. In this approach, a Neumann boundary condition can be applied to an infinite planar reflector by reflecting the sound source about the reflector, resulting in a secondary *image source*. This approach can be recursively applied to model multiple reflections.

The image source method is an exact solution of the wave equation for a rectangular room with perfectly rigid, perfectly smooth walls. The method models perfectly specular (mirror-like) reflections only. The method does not account for general, angle-dependent impedances. It does not model surface scattering from non-smooth surfaces. And most importantly, when applied to finite planar reflectors in non-rectangular domains [12], it does not account for diffraction and scattering effects caused by the finite extent of the reflectors. Nonetheless, it remains a popular method in many applications such as architectural acoustics.

Recent improvements to the image source method are based on the fact that image sources only need to be generated for surfaces that are *visible* to the source [17]. Applying this observation recursively allows the set of image sources to be organized into an *image source tree* (or *visibility tree*). Efficient visibility algorithms can then be used to generate the image source tree, resulting in improved efficiency.

2.3.2 Stochastic Ray Tracing

A more general method for using rays to simulate sound propagation is *stochastic ray tracing* [37, 88, 46]. In this approach, a large number of rays are traced from

the source. These rays all carry equal amounts of *energy*, such that the total energy carried by all rays is proportional to the intensity of the source. As the rays propagate through the domain, they are reflected upon encountering the boundaries. The method keeps track of the energy lost at each reflection due to absorption, as well as the total distance traveled by each ray. Rays are counted as they pass through a *detection sphere* around the listener position. The energy carried and distance traveled by each ray is then used to determine the variation of acoustic energy over time. This information can then be used to estimate the pressure at the listener position [40].

Surface scattering can also be modeled using stochastic ray tracing. This is achieved using a *random incidence scattering coefficient* [89], which, for any given boundary point, models the probability that an incident ray is scattered *away* from the specular reflection direction. This can be used to generate reflected rays traveling along randomly chosen directions, thereby modeling *diffuse reflections* or general surface scattering.

2.3.3 Volume Tracing

Tracing rays can lead to errors in sampling the domain boundary. Moreover, choosing an appropriate size for the detection sphere at the listener remains a tricky problem [78]. An alternative approach is to trace *volumes* such as cones, pyramids [24] or frusta [43]. In these approaches, testing whether a propagation path reaches the listener requires a simple containment test with the corresponding cone, pyramid or frustum. Adaptive frustum sampling approaches have also been developed [18], enabling frustum tracing to achieve near-interactive performance for early reflections, even on complex models.

2.3.4 Diffraction

Neither the image source method nor stochastic ray tracing accounts for diffraction around obstacles. Diffraction is a low-frequency phenomenon, and is not modeled by high-frequency geometric acoustics techniques. Therefore, several techniques have been developed to augment geometric acoustics by explicitly computing the effects of diffraction. These methods begin by identifying *diffracting edges*, i.e., edges around which sound waves bend [79]. A suitable edge diffraction model is then applied to simulate diffraction, as well as the interactions between diffraction and reflecting surfaces. Two commonly-used diffraction models are briefly discussed below.

Uniform Theory of Diffraction The Uniform Theory of Diffraction (UTD), originally developed for electromagnetic waves [36], is used to model diffraction about an infinite, perfectly rigid wedge. This makes it inaccurate for the finite diffracting edges encountered in all finite domains, but the relative simplicity of the model makes it attractive, especially for interactive applications [85].

Biot-Tolstoy-Medwin model The Biot-Tolstoy-Medwin (BTM) model [76] is based on the Huygens principle. Diffraction from a finite wedge is modeled by placing infinitesimal secondary sources along the diffracting edge, and integrating the contributions due to all of them. This allows diffraction to be accurately modeled, but the approach does not easily scale to multiple orders of diffraction, or to complex scenes with many diffracting edges, since each diffracting edge must be densely sampled. Recent work in this area has focused on these two aspects. From-region visibility algorithms have been applied to efficiently cull invalid diffraction paths [7], allowing BTM to handle large, complex scenes. Monte Carlo methods have been used to efficiently evaluate the BTM integrals using GPUs [56].

2.3.5 Acoustic Rendering Equation

Many of the above techniques can be unified into a single formulation using the *acoustic rendering equation* [67]. This is an integral equation based on a transport theory formulation of the propagation of acoustic energy. We use this equation to formulate the acoustic transfer operators described in Chapter 3. Note that since this is an energy-based formulation, it cannot model detailed variations in the phase of the pressure field.

2.4 Impulse Responses

During simulation (and particularly in interactive applications), we often need to perform simulations in the same environment, with no changes to the scene, but with different sound signals. In such situations, instead of repeating the entire simulation, we exploit the fact that acoustics is a *linear, time-invariant* system [38].

Let $f_1(t)$ and $f_2(t)$ be two sound signals that can be emitted by a given source in a given environment. Let $g_1(t)$ and $g_2(t)$ be the corresponding signals received at a given listener position. Let $\mathcal{H}$ be the sound propagation operator such that $g_1 = \mathcal{H}(f_1)$ and $g_2 = \mathcal{H}(f_2)$. Then $\mathcal{H}$ is *linear* if $\mathcal{H}(f_1 + f_2) = g_1 + g_2$, and $\mathcal{H}(\alpha f_1) = \alpha g_1$ (and similarly for f_2). $\mathcal{H}$ is *time-invariant* if $\mathcal{H}(f_1(t - \Delta t)) = g_1(t - \Delta t)$ (and similarly for f_2). This property is also referred to as *shift-invariance*.

Since the differential operator is linear time-invariant, it can be shown that the propagation of sound waves, which is governed by a differential equation, can be described as a linear time-invariant system. Then, like any linear time-invariant system, sound propagation can be completely characterized by its *impulse response*, i.e., $h(t) = \mathcal{H}(\delta(t))$, where $\delta(t)$ is the Dirac delta function:

$$\delta(t) = \begin{cases} +\infty & \text{if } t = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.10}$$

or, when dealing with discrete functions, the Kronecker delta function:

$$\delta[t] = \begin{cases} 1 & \text{if } t = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

Simulation is used to determine the impulse response. The actual signal received at the listener for any given source signal f is then obtained through *convolution*:

$$g = h \star f = \int_{-\infty}^{\infty} h(t - \tau) f(\tau) d\tau. \tag{2.12}$$

Note that impulse responses for physically valid sound propagation must be *causal*, i.e., $h(t) = 0$ for $t < 0$.

2.4.1 Frequency Responses

Frequency-domain algorithms (such as FEM and BEM) compute the *frequency response*, which is related to the impulse response through the Fourier transform:

$$H(\omega) = \int_{-\infty}^{\infty} h(t) e^{-\iota\omega t} dt, \tag{2.13}$$

where H is the frequency response, expressed as a function of angular frequency. The impulse response can be recovered from the frequency response using the inverse Fourier transform:

$$h(t) = \int_{-\infty}^{\infty} H(\omega) e^{\iota\omega t} d\omega. \tag{2.14}$$

The frequency response is the result of applying the sound propagation operator to the Fourier transform of the delta function, i.e., $H(\omega) = \mathcal{H}(\delta(\omega))$. The Fourier coefficients of δ are all equal to 1.

Essentially, the Fourier transform expresses an arbitrary time signal as a weighted sum of complex-valued sinusoids. In general, the values of $H(w)$ are complex numbers, containing both magnitude and phase:

$$H(\omega) = A_\omega e^{\iota\phi_\omega}. \tag{2.15}$$

Scaling the magnitudes changes the frequency content of a signal. For example, increasing the magnitudes of the low-frequency terms in a Fourier expansion results in a bass boost. Scaling the *phase* terms, however, alters the temporal characteristics of the signal. For example, multiplying $H(\omega)$ with $e^{-\iota\omega\Delta t}$ (where $\iota = \sqrt{-1}$) has the effect of delaying the signal by Δt:

$$\int_{-\infty}^{\infty} H(\omega)e^{-\iota\omega\Delta t}e^{\iota\omega t}d\omega = \int_{-\infty}^{\infty} H(\omega)e^{\iota\omega(t-\Delta t)}d\omega = h(t-\Delta t). \tag{2.16}$$

In this manner, the Fourier transform allows both scaling and delays to applied to the acoustic response using a uniform representation of both kinds of operations.

2.4.2 Echograms

Energy-based simulation algorithms, such as stochastic ray tracing, do not compute the impulse response. Instead, they compute the acoustic energy received by the listener as a function of time, often referred to as the *echogram*. We denote the pressure at the listener by $p(t)$, and the echogram by $e(t)$. Moreover, we denote the corresponding Fourier transforms by $P(\omega)$ and $E(\omega)$, respectively. Given the echogram computed by the simulation, $e(t)$ or $E(\omega)$, the impulse response, $p(t)$ or

$P(\omega)$, can be computed as follows.

First, we note that due to the definition of acoustic energy, $|E(\omega)| \propto |P(\omega)|^2$. This allows us to use a square root to compute the magnitude of the frequency response. The phase information is irretrievably lost due to the energy-based nature of the simulation. This information may be faked, however, using random phase, a minimum-phase assumption, or by simply copying the phase of $E(\omega)$ [40]. Once both the magnitude and phase are available, the impulse response can be computed through an inverse Fourier transform.

2.5 Auralization

In interactive applications, sound propagation simulation is used to compute an impulse response from a source to a listener. This is in turn used to generate immersive audio with acoustic effects, using convolution. For example, an impulse response can add reverberation to a cathedral, or occlusion effects behind a pillar. This process of using the results of sound propagation simulation to generate audio signals that incorporate acoustic effects is sometimes referred to as *auralization*. The basic idea is to use convolution with a *dry* or *anechoic* audio clip (i.e., one which does not contain any propagation effects) to generate the audio clip heard by the listener.

Performing convolution by evaluating the integral in equation 2.12 is computationally expensive. Therefore, most applications use the Fourier transform to perform convolution, due to the following property:

$$\mathcal{F}(f \star g) = \mathcal{F}(f) \cdot \mathcal{F}(g), \tag{2.17}$$

i.e., convolution in the time domain is multiplication in the frequency domain, and vice-versa. Using the fast Fourier transform algorithm, the above equation can

be used to efficiently perform convolution.

2.5.1 Real-Time Convolution

In many interactive applications, the source signal may be too long for even the Fourier-transform-based convolution to be efficient. Moreover, in many cases, the source and/or listener may be moving, causing the impulse response to change over time even as the source signal is being emitted. For such situations, the *short-time Fourier transform* (STFT) [3] is used to perform real-time convolution.

The source signal is divided into sequential *frames* of a short duration, say 100 ms. Each frame is convolved with the impulse response, with the result being longer than the frame duration. For example, if a 100 ms frame is convolved with a 1s impulse response, the result is a 1.1s signal. As each frame is convolved, the results are combined together to form the final output audio stream. Two common approaches to perform this combination are *overlap-add* and *overlap-save* [3].

Note that since each frame may be convolved with a different IR, the effects of time-varying impulse responses can be taken into account. However, as IRs may change unpredictably from frame to frame, audible artifacts may occur between frames. These can be alleviated using carefully chosen *windowing* filters for interpolation between frames.

2.5.2 Spatialization

For immersive sound rendering in interactive applications, it is important to model the *directional* nature of sound. In other words, a user should be able to perceive the direction from which sound from the source reaches the listener. This process is sometimes referred to as *spatialization*. This is often achieved by decomposing

the sound field into a superposition of point sources or plane waves, whose sound arrives at the listener from a fixed direction. Spatialization can be performed by determining what signals to output from the user's speakers so as to reproduce the effect of the elementary point sources or plane waves.

Amplitude Panning Amplitude panning [60] refers to the general approach of using the direction of the source (or plane wave) relative to the listener, as well as the direction of a speaker relative to the listener, to compute the signal to emit from the speaker. This is essentially described as a real-valued scaling factor that is applied to the source signal. This way, for each speaker, a *panning weight* or *panning gain* is computed, and applied to the source signal. There are multiple ways to define panning weights, each with its own pros and cons.

Ambisonics Ambisonics [50] refers to an approach for representing the sound field due to a plane wave or point source using spherical harmonics (SH). Essentially, the SH coefficients of the sound field due to a plane wave are stored as separate channels in an audio file. (This format is also called *B-format*.) A hardware decoder is typically used to reconstruct a multichannel audio stream from the SH coefficients. Ambisonics have the advantage that the same audio signal can be used to automatically scale to any arbitrary speaker system.

2.5.3 Binaural Rendering

All of the above spatialization techniques (amplitude panning, ambisonics, etc.) are based on measuring or simulating the sound field at a single point. Humans, on the other hand, have two ears, which enables sound sources to be located more accurately. Moreover, the interactions between the head, ears and the sound field

results in subtle differences between the sound received at each ear [10]. These effects are all captured using the *head-related transfer function* (HRTF), or its time-domain equivalent, the *head-related impulse response* (HRIR) [10].

Given a point source or plane wave incident at the listener along a direction $\mathbf{s}$, we define two HRTFs, one for each ear, denoted by $HRTF_L(\mathbf{s})$ and $HRTF_R(\mathbf{s})$. These are used as follows. We first measure or simulate the sound field at the center of the head, in the absence of the head, and denote this by $F(\omega)$. The signals received at the left and right ears are then denoted by $G_L(\omega)$ and $G_R(\omega)$, respectively, and given by:

$$G_L(\omega) = HRTF_L(\mathbf{s})F(\omega), \tag{2.18}$$

$$G_R(\omega) = HRTF_R(\mathbf{s})F(\omega). \tag{2.19}$$

The signals G_L and G_R are then played on the two channels of a headphone system. This gives the user the experience of actually being in the virtual environment. Note that this approach cannot work with general speaker systems unless special filtering is performed to cancel the effects of room acoustics in the listening space, as well as the fact that all speakers can be heard at both ears. This process is referred to as *crosstalk cancellation* [10]. Also note that for maximum impact, each user's head should be used to compute a custom HRTF. However, this is usually impractical, so standard measurements [28] are typically used.

2.6 Precomputed Sound Propagation

In interactive applications, the goal is to model sound propagation in scenarios where the listener, the sound sources, and in some cases even the geometry may be moving. In such situations, running detailed numerical simulations, or performing compute-intensive ray tracing every time either the source, listener, or geometry move is usually impractical. For many interactive applications, however, we may assume that large portions of the geometry are static. We may then precompute impulse responses between static portions of the scene and use them to efficiently reconstruct impulse responses for a moving listener. Such precomputation-based methods may be broadly classified depending on whether or not they assume the source to be static as well.

2.6.1 Static Source Methods

Beam tracing [27, 42] is one example of a precomputation-based method that assumes static sources. Given a source position, it is possible to generate image sources to model reflected (or even diffracted) sound. The beam tracing method generates *beams* (or frusta) which correspond to each image source. A listener receives sound from an image source if and only if it lies inside the corresponding beam. This allows a *beam tree* to be precomputed in an offline process. During interactive simulation, the listener position is used to efficiently determine which beams the listener lies inside, and this information is used to render sound from the corresponding image sources.

Alternatively, the domain boundary may be divided into several *patches*. Impulse responses can be computed from the source to a point on each patch. During interactive simulation, the points on each patch are treated as point sources emitting

a signal given by the corresponding precomputed impulse response. Contributions from each patch are added together to obtain the final impulse response at the current position of a moving listener. This approach has been combined with the acoustic rendering equation to obtain an efficient frequency-domain method [68] for precomputing sound propagation from a static source.

2.6.2 Moving Source Methods

In many interactive applications, the scene can naturally be divided to *cells* interconnected by *portals* [25]. In such cases, it is possible to precompute impulse responses from the center of each cell to the center of the same cell, as well as to the centers of each of its neighboring cells [72]. During interactive simulation, the positions of the moving source and moving listener are used to determine the cells they are in. Paths are then computed from the source cell to the listener cell in the cell-and-portal graph, and impulse responses for each consecutive pair of cells along such a path are convolved sequentially to determine the final impulse response. Compact representations have been developed for such approaches [83], but these are limited to scenes which can be decomposed into cells and portals. This may not be possible for many common kinds of environments, such as outdoor environments.

Numerical methods have also been used to develop precomputation-based sound propagation algorithms. Recently [63], ARD was used to simulate sound propagation from a grid of sources distributed throughout an environment, to a grid of listeners also distributed throughout the environment. During interactive simulation, the nearest grid points to the source and listener are determined, and interpolation is performed to determine the source-to-listener impulse response given the precomputed impulse responses between the grid points. While this method has high performance, it requires very large amounts (several gigabytes) of storage for the pre-

computed data, making it impractical for interactive applications on consumer-grade hardware.

2.7 Statistical Models

As the preceding discussion has shown, direct application of numerical methods tends to be impractical for interactive applications. Geometric methods are not fast enough for interactive simulation of higher-order reflections and reverberation, since this would require very high orders of reflection, and a very large number of rays to be traced. While precomputed methods can be used for this technique, the size of the precomputed data is often an issue. Therefore, most interactive applications model effects like reverberation using simple statistical models.

The *Eyring model* [23] is an example of a statistical model for reverberation. It represents reverberation using an exponentially decaying impulse response, modulated by a noise function. This approximation has been developed for single rectangular rooms, but is often used for arbitrary environments. Recently, more sophisticated models have been developed for coupled rooms [75]. Reverberation models are often expressed using recursive *infinite impulse response* (IIR) filters, and implemented using feedback delay networks [35]. Such artificial reverberation models form the basis of the Interactive 3D Audio Level 2 specification [33], which is used in most modern games.

Chapter 3

Acoustic Transfer Operators

In this chapter, we describe our formulation of *acoustic transfer operators*. We develop acoustic transfer operators based on the acoustic rendering equation, and outline a family of algorithms collectively called *Precomputed Acoustic Radiance Transfer* (PART). The basic approach involves precomputing acoustic transfer operators and using them for efficient interactive sound propagation. Two such algorithms are described later in Chapters 3.3.3 and 5.

3.1 Acoustic Rendering Equation

The acoustic transfer operator is derived from an integral equation describing the distribution of acoustic energy in an environment. We begin by defining the quantities required to describe such energy distributions.

3.1.1 Acoustic Energy Transport

The energy contained in a sound field at a point is measured as the product of pressure p and particle velocity $\mathbf{u}$. The resulting "energy field" is a vector quantity, and can

be analyzed using the techniques of transport theory. To perform the analysis, we define a *phonon* as a quantum of acoustic energy [11]. Phonons are virtual particles that carry acoustic energy (in a sense analogous to photons), and travel in straight lines.

Flux The total amount of energy (or equivalently, the total number of phonons) passing through a region of space with area A is called the *flux* through the region. Consider a differential volume element $d\mathbf{r}$ around a point $\mathbf{r}$, and differential solid angle $d\mathbf{s}$ around the direction $\mathbf{s}$. The net number of phonons traveling through the boundary of $d\mathbf{r}$ within the range of solid angles $d\mathbf{s}$ is called the *flux density* $\Phi(\mathbf{r}, \mathbf{s})$.

Irradiance Consider a surface point $\mathbf{r}$. The flux striking a differential area $d\mathbf{A}$ from a direction $\mathbf{s}$ is called the *irradiance*:

$$E(\mathbf{r}, \mathbf{s}) = \frac{d\Phi(\mathbf{r}, \mathbf{s})}{d\mathbf{A} \cdot \mathbf{s}}. \tag{3.1}$$

Note that area is represented as a vector quantity, whose direction is equal to the normal of the surface patch. The total irradiance at a point can be obtained by integrating the above quantity over all directions in the hemisphere around the surface normal.

Radiance The flux entering or leaving a differential area $d\mathbf{A}$ from a direction $\mathbf{s}$ per unit solid angle is called the *radiance*:

$$L(\mathbf{r}, \mathbf{s}) = \frac{d^2\Phi(\mathbf{r}, \mathbf{s})}{(d\mathbf{A} \cdot \mathbf{s})d\mathbf{s}}. \tag{3.2}$$

Radiance arriving at a point $\mathbf{r}$ is called *incident* radiance, whereas radiance leaving a point $\mathbf{r}$ is called *exitant* radiance.

Radiance and Echograms Note that all of the above quantities – flux density, irradiance, and radiance – are functions of time. The time variable t has been hidden in the above equations for simplicity. If the flux density distribution in an environment is induced by a sound source emitting a unit impulse, then the echogram at the listener position can be obtained by integrating the radiance at the listener position over the unit sphere (i.e., over all directions).

Therefore, we need a way to compute the radiance at any point in the environment, accounting for various sound propagation phenomena. This is achieved using the acoustic rendering equation.

3.1.2 The Acoustic Rendering Equation

The acoustic rendering equation is an integral equation which governs the exitant acoustic radiance at any surface point in the scene:

$$L(\mathbf{r},\mathbf{s},t) = L_0(\mathbf{r},\mathbf{s},t) + \int_{\partial\Omega} V(\mathbf{r},\mathbf{r}')G(\mathbf{r},\mathbf{r}')\rho(\mathbf{s}',\mathbf{s},\mathbf{r},t) \star P(\mathbf{r},\mathbf{r}',t) \star L(\mathbf{r}',\mathbf{s}',t)d\mathbf{r}'. \quad (3.3)$$

The integration is carried out over all surface points $\mathbf{r}'$, and $\mathbf{s}'$ is the direction from $\mathbf{r}'$ to $\mathbf{r}$. This is the time-domain version of the acoustic rendering equation. In the frequency-domain version, the time variable t is replaced by the frequency ν, and the convolutions are replaced by multiplications. L_0 is the *direct* radiance, i.e., the exitant radiance at point $\mathbf{r}$, along direction $\mathbf{s}$, reflected directly from the sound source. The definitions of the various terms in the integrand are provided below.

Visibility The visibility function $V(\mathbf{r},\mathbf{r}')$ describes whether the points $\mathbf{r}$ and $\mathbf{r}'$ are mutually visible. Its value is 1 if the points are mutually visible, and 0 otherwise.

The visibility function is typically evaluated using a ray tracer.

Form Factor The form factor, or geometry term $G(\mathbf{r}, \mathbf{r}')$ accounts for the relative positions and orientations of differential surface patches $d\mathbf{A}$ and $d\mathbf{A}'$ around the points $\mathbf{r}$ and $\mathbf{r}'$, respectively:

$$G(\mathbf{r}, \mathbf{r}') = \frac{\cos\theta_{in} \cos\theta_{out}}{|\mathbf{r} - \mathbf{r}'|^2}, \tag{3.4}$$

where θ_{in} is the angle between $d\mathbf{A}$ and $\mathbf{s}'$, and θ_{out} is that between $d\mathbf{A}'$ and $\mathbf{s}'$.

Bidirectional Reflectance Distribution Function The bidirectional reflectance distribution function (BRDF) $\rho(\mathbf{s}', \mathbf{s}, \mathbf{r}, t)$ is the ratio of exitant radiance along $\mathbf{s}$ and incident irradiance along $\mathbf{s}'$, at the point $\mathbf{r}$:

$$\rho(\mathbf{s}', \mathbf{s}, \mathbf{r}, t) = \frac{dL(\mathbf{r}, \mathbf{s}, t)}{dE(\mathbf{r}, \mathbf{s}', t)}. \tag{3.5}$$

In the time domain, the BRDF is a function of time. Intuitively, this is because the surface at point $\mathbf{r}$ may vibrate in a complex manner when a sound wave is incident on it. As a result, even through the incident sound wave may be an impulse, the reflected sound wave may have a more complex temporal structure. In the frequency domain, the BRDF is defined for multiple frequencies. The phase terms of the BRDF values account for the complex temporal structure of reflected sound waves.

Propagation Term The propagation term $P(\mathbf{r}, \mathbf{r}', t)$ accounts for propagation of phonons from $\mathbf{r}'$ to $\mathbf{r}$. It is used to model propagation delays and air absorption:

$$P(\mathbf{r}, \mathbf{r}', t) = e^{-\alpha|\mathbf{r}-\mathbf{r}'|} \delta\left(t - \frac{|\mathbf{r} - \mathbf{r}'|}{c}\right), \tag{3.6}$$

where α is the air absorption coefficient, and c is the speed of sound.

3.2 Transfer Operators

The acoustic rendering equation can be rewritten in the following way:

$$L(\mathbf{r},\mathbf{s},t) = L_0(\mathbf{r},\mathbf{s},t) + \int_{\partial\Omega} R(\mathbf{r},\mathbf{s},\mathbf{r}',t) \star L(\mathbf{r}',\mathbf{s}',t)d\mathbf{r}', \tag{3.7}$$

where $R(\mathbf{r},\mathbf{s},\mathbf{r}',t) = V(\mathbf{r},\mathbf{r}')G(\mathbf{r},\mathbf{r}')\rho(\mathbf{s}',\mathbf{s},\mathbf{r},t) \star P(\mathbf{r},\mathbf{r}',t)$ is the *reflection kernel*. Equations of the above form are called *Fredholm equations of the second kind*. Such equations have a unique, continuous solution which can be written as a *Liouville-Neumann series* as shown below:

$$L(\mathbf{r},\mathbf{s},t) = \sum_{i=0}^{\infty} L_i(\mathbf{r},\mathbf{s},t), \tag{3.8}$$

with the individual elements of the series, L_i, related by the following recursive relation:

$$L_{i+1}(\mathbf{r},\mathbf{s},t) = \int_{\partial\Omega} R(\mathbf{r},\mathbf{s},\mathbf{r}',t) \star L_i(\mathbf{r}',\mathbf{s}',t)d\mathbf{r}'. \tag{3.9}$$

Intuitively, L_i is the radiance distribution after i orders of reflection, and the recursive relation above calculates the $i+1$ order radiance by reflecting the i order radiance at the domain boundary.

In operator notation, the above equations can be written as:

$$
\begin{aligned}
\mathcal{L}_{i+1} &= \mathcal{R}\mathcal{L}_i, \\
\mathcal{L} &= \sum_{i=0}^{\infty} \mathcal{L}_i \\
&= \sum_{i=0}^{\infty} \mathcal{R}^i \mathcal{L}_0 \\
&= \mathcal{T}\mathcal{L}_0, \qquad (3.10)
\end{aligned}
$$

where $\mathcal{T} = (\mathcal{I} - \mathcal{R})^{-1}$ is called the *acoustic transfer operator*. Since $||\mathcal{R}|| < 1$, where $||\mathcal{R}||$ denotes the norm of the operator $\mathcal{R}$, we have $\mathcal{T} = (\mathcal{I} + \mathcal{R} + \mathcal{R}^2 + \cdots)$ [86].

As the above equations show, applying the acoustic transfer operator to the direct radiance distribution yields the final radiance distribution. More importantly, for a static domain boundary, the transfer operator can be precomputed and efficiently applied to the direct radiance at run-time, resulting in efficient simulation of detailed, high-order sound propagation effects.

3.3 Discrete Transfer Operators

For transfer operators to be used in a practical application, they must first be converted to a discrete representation. Essentially, this involves discretizing the independent variables $\mathbf{r}$, $\mathbf{s}$, and t. This process is described below.

Surface Discretization The domain boundary $\partial\Omega$ is discretized by subdividing it into a number of *patches*. The patches are analogous to the surface elements used in boundary element methods. The radiance functions are assumed to be constant over each patch. The value of radiance in a patch is obtained by computing the radiance at a single *surface sample* point (or *collocation point*) associated with each patch.

Direction Discretization We assume that the transfer operator and the radiances $\mathcal{L}_0$ and $\mathcal{L}$ are constant over all directions. In other words, the transfer operator is independent of direction $\mathbf{s}$. The implications of this assumption on the kinds of acoustic phenomena that can be modeled by the transfer operator are discussed in more detail in Chapters 3.3.3 and 5.

Time Discretization The time variable t is sampled at the audio sampling rate (e.g., 44.1 kHz or 48 kHz) so as to capture acoustic effects across the audible range of frequencies.

3.3.1 Matrix Representation

After performing surface, direction, and time discretization, the transfer operator can be represented as a matrix, as described below.

Radiance Vectors Suppose the number of surface samples is n_r and the number of time samples is n_t. Then the acoustic radiance can be written as an $n_r n_t \times 1$ vector, which, in block notation, is:

$$\mathbf{l} = \begin{bmatrix} \mathbf{l}(1) \\ \hline \mathbf{l}(2) \\ \hline \vdots \\ \hline \mathbf{l}(n) \end{bmatrix}, \tag{3.11}$$

where $\mathbf{l}(1)$ is $n_t \times 1$ vector representing the radiance at surface sample 1, etc.

Convolution Matrices Consider a discrete signal l_j with n_t samples, represented as an $n_t \times 1$ vector, and an impulse response (or other discrete filter) h with n_t samples. The convolution $g_j = h \star l_j$ can be represented as a matrix-vector product

$\mathbf{g}_j = \mathbf{H}\mathbf{l}_j$. For example, with an impulse response with 3 non-zero samples:

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & h_1 & h_2 & h_3 & 0 & 0 & 0 \\ 0 & 0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & h_1 & h_2 & h_3 & 0 \\ 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 \end{bmatrix}. \tag{3.12}$$

Note that the signal l_j must be sufficiently zero-padded to prevent aliasing.

Block Matrix Representation The radiance at surface sample j can be written in terms of the direct radiance at surface samples j' as follows:

$$\mathbf{l}(j) = \sum_{j'=0}^{n_r-1} \mathbf{T}(j, j')\mathbf{l}_0(j'), \tag{3.13}$$

where $\mathbf{T}(j, j')$ is a convolution matrix describing the acoustic transfer from sample j' to sample j. In block matrix notation, the transfer operator has the following form:

$$\mathbf{T} = \left[\begin{array}{c|c|c|c} \mathbf{T}(1,1) & \mathbf{T}(1,2) & \cdots & \mathbf{T}(1,n) \\ \hline \mathbf{T}(2,1) & \mathbf{T}(2,2) & \cdots & \mathbf{T}(2,n) \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline \mathbf{T}(n,1) & \mathbf{T}(n,2) & \cdots & \mathbf{T}(n,n) \end{array}\right]. \tag{3.14}$$

In this form, the transfer matrix has dimensions $n_r n_t \times n_r n_t$.

Frequency-Domain Representation To obtain a frequency-domain representation, we take the Fourier transform of the radiance vectors and the transfer matrix:

$$\mathbf{F}\mathbf{l} = \mathbf{F}\mathbf{T}\mathbf{F}^{-1}\mathbf{F}\mathbf{l}_0, \tag{3.15}$$

where $\mathbf{F}$ is the matrix representation of the Fourier transform operator. Since the Fourier basis functions (i.e., complex sinusoids) are eigenfunctions of linear time-invariant operators, applying the Fourier transform to a convolution matrix converts it to a diagonal matrix. Therefore, each block $\mathbf{T}(i,j)$ becomes a diagonal matrix after applying the Fourier transform. This allows the transfer operator to be split into multiple independent matrices, once for each frequency sample, resulting in n_t matrices, each of dimension $n_r \times n_r$.

3.3.2 Alternative Derivation

An alternative way of deriving transfer matrices from the acoustic rendering equation is follows. Suppose the boundary $\partial\Omega$ is discretized into n patches, numbered 0 through $n-1$. Suppose the patch with index i is denoted by $\partial\Omega_i$, and the corresponding sample point is denoted by $\mathbf{r}_i$. Using the acoustic rendering equation to calculate the exitant radiance at $\mathbf{r}_i$ yields:

$$L(\mathbf{r}_i, t) = L_0(\mathbf{r}_i, t) + \int_{\partial\Omega} R(\mathbf{r}_i, \mathbf{r}', t) \star L(\mathbf{r}', t) d\mathbf{r}'. \tag{3.16}$$

For any point $\mathbf{r}' \in \partial\Omega_j$, we have $L(\mathbf{r}', t) = L(\mathbf{r}_j, t)$, since radiance is assumed to be constant over each patch, and $\mathbf{r}_j \in \partial\Omega_j$. This allows us to split the integral as follows:

$$L(\mathbf{r}_i, t) = L_0(\mathbf{r}_i, t) + \sum_{j=0}^{n-1} L(\mathbf{r}_j, t) \star \left(\int_{\partial\Omega_j} R(\mathbf{r}_i, \mathbf{r}', t) d\mathbf{r}' \right). \tag{3.17}$$

We define:

$$R_{i,j}(t) = \int_{\partial\Omega_j} R(\mathbf{r}_i, \mathbf{r}', t) d\mathbf{r}'. \tag{3.18}$$

This allows us to rewrite Equation 3.17 as:

$$L(\mathbf{r}_i, t) = L_0(\mathbf{r}_i, t) + \sum_{j=0}^{n-1} L(\mathbf{r}_j, t) \star R_{i,j}(t). \tag{3.19}$$

The functions $R_{i,j}(t)$ may be precomputed and stored in convolution matrix form, denoted by $\mathbf{R}(i, j)$. These convolution matrices may be assembled in the block matrix form discussed in Section 3.3.1, yielding the *one-bounce transfer matrix*, $\mathbf{R}$. The acoustic transfer operator is then obtained as:

$$\begin{aligned} \mathbf{T} &= \mathbf{I} + \mathbf{R} + \mathbf{R}^2 + \cdots \\ &= (\mathbf{I} - \mathbf{R})^{-1}. \end{aligned} \tag{3.20}$$

3.3.3 Complexity

This section describes the asymptotic time and storage complexity of using transfer operators.

Time Complexity Applying the transfer operator to a direct radiance distribution is a matrix-vector multiplication. The complexity of a matrix-vector multiplication is $O(n^2)$, for matrices of dimension $n \times n$ and vectors of dimension $n \times 1$. For a time-domain transfer operator, the time complexity is therefore $O(n_r^2 n_t^2)$. For frequency-domain transfer operators, n_t independent matrix-vector multiplications result in a time complexity of $O(n_t n_r^2)$.

Storage The storage cost of time-domain as well as frequency-domain transfer operators is $O(n_t n_r^2)$. For frequency-domain transfer operators, this is because there are n_t independent matrices, each of dimension $n_r \times n_r$. For time-domain transfer operators, since each $n_t \times n_t$ convolution matrix contains only n_t unique numbers,

the storage complexity is only $O(n_t n_r^2)$.

In practice, n_r and n_t can have large values, and storing and using transfer operators in a naive manner is often impractical, requiring several gigabytes of storage. In the following chapters, we will describe efficient ways to store and use acoustic transfer operators.

Chapter 4

Frequency-Domain Acoustic Transfer Operators

In this chapter, we discuss a specific algorithm based on the PART framework, for simulating higher-order diffuse reflections of sound [8]. The algorithm performs its calculations in the frequency domain. We also describe an approach for incorporating higher-order reflections in the transfer operator without performing multi-bounce ray tracing.

4.1 Domain Discretization

The algorithm represents acoustic radiance and acoustic transfer operators by discretizing in the spatial and frequency domains as discussed below.

4.1.1 Echogram Representation

Echograms are discretized into N time-domain samples, where the value of N is chosen based on the desired audio sampling frequency and the length of the IR modeled (which could be tuned based on the expected reverberation time of a room). We compute the Discrete Fourier Transform (DFT) using the Fast Fourier Transform (FFT) algorithm. A unit impulse emitted by the source at time t = 0 has all

Fourier coefficients set to 1. Since the Fourier transform is linear, attenuation and accumulation of IRs can be performed easily (n denotes a discrete sample index):

$$\mathcal{F}(af_1(n) + bf_2(n)) = a\mathcal{F}(f_1(n)) + b\mathcal{F}(f_2(n)). \tag{4.1}$$

Unlike in the time domain, in the frequency domain delays can also be applied using a scale factor, since the Fourier basis vectors are eigenvectors of linear time-invariant operators:

$$\mathcal{F}(f(n - \Delta n)) = e^{-\iota\omega\Delta n}\mathcal{F}(f(n)). \tag{4.2}$$

Note that care must be taken to ensure that the delays align on time-domain sample boundaries, otherwise the inverse Fourier transform will contain non-zero imaginary parts.

Given the length of the impulse response to be modeled, we truncate the Fourier coefficients of the echogram, retaining a (relatively) small number of coefficients. This gives a discrete representation for acoustic radiance at a point.

Echogram Reconstruction Computing the echogram using the above expressions for delay and attenuation results in a frequency-domain signal, with a limited set of Fourier coefficients. Before recovering an impulse response for convolution, the missing Fourier coefficients of the echogram must be reconstructed. This is achieved by replicating the Fourier coefficients in the frequency domain until the desired number of samples are obtained. In the time-domain, this operation is equivalent to upsampling the signal.

4.1.2 Surface Sampling

We parameterize the scene surface by mapping the primitives to the unit square (a *uv* texture mapping) using Least Squares Conformal Mapping (LSCM) [47]. The user specifies the texture dimensions; each texel of the resulting texture is mapped to a single surface sample using an inverse mapping process. The number of texels mapped to a given primitive is weighted by the area of the primitive, to ensure a roughly even distribution of samples. We chose the LSCM algorithm for this purpose since it is widely used in current 3D modeling tools (e.g., Blender [1]); it can be replaced with any other technique for sampling the surfaces as long as the number of samples generated on a primitive is proportional to its area.

Diffuse Patches Since we are modeling diffuse reflections, it is not necessary to model directional variation of acoustic radiance at surface points. Therefore, we assume that all surface patches are diffuse.

4.2 Precomputation

Our algorithm provides two main improvements over the state-of-the-art acoustic radiance transfer algorithms: (a) we decouple the source position from the precomputed data by computing an *acoustic transfer operator*; and (b) we use the SVD to compress the transfer operator and quickly compute higher-order reflections. The rest of this chapter details how our algorithm achieves these improvements over the state-of-the-art. Our overall approach is as follows (see Figure 4.1):

- **Preprocessing.** We sample the surface of the scene and compute a transfer operator which models one or more orders of diffuse reflections of sound among

[1] `http://www.blender.org`

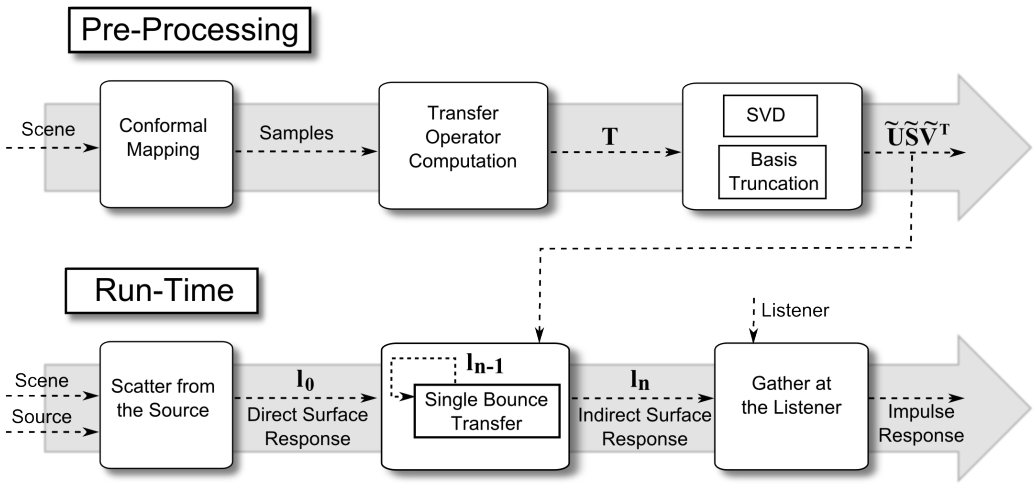


Figure 4.1: Overview of our algorithm. In a precomputation step, we sample the surfaces on the scene, and compute a one-bounce transfer operator for these samples (**T**). We then use the SVD to compute the modes of the transfer operator. At runtime, we shoot rays from the source (which may move freely) and compute direct IRs at the surface samples. We then apply the transfer operator (with a user-specified number of modes retained) repeatedly to quickly obtain the multi-bounce indirect IRs at the surface samples. We compute the IR at the listener position in a final gathering step.

the surface samples.

- **Run-time.** We compute first-order diffuse reflections at run-time, and apply the transfer operator to efficiently compute higher-order diffuse reflections.

4.2.1 Transfer Operator Computation

The acoustic transfer operator is expressed over a set of p samples chosen over the surface of the scene. Let there be f Fourier coefficients per surface sample. All subsequent computations are performed on each Fourier coefficient independently.

For each frequency ω_m, we define the acoustic radiance vector $\mathbf{l}(\omega_m)$, which contains p elements that represent the m^{th} Fourier coefficients of the radiance at each surface sample. For the sake of brevity, we shall omit the parameter ω_m from the

equations in the rest of this chapter as it may be obvious from the context.

The Neumann series expansion of the acoustic rendering expressed in matrix form is:

$$\mathbf{l}_{n+1}(\omega_m) = \mathbf{R}(\omega_m)\mathbf{l}_n(\omega_m), \tag{4.3}$$

where $\mathbf{l}_n(\omega_m)$ contains the m^{th} Fourier coefficients of the radiance at each surface sample after n reflections. The *reflection matrix* $\mathbf{R}(\omega_m)$ models the effect of one diffuse reflection. The $(i,j)^{th}$ element of $\mathbf{R}(\omega_m)$ describes how the m^{th} Fourier coefficient at surface sample j affects the m^{th} Fourier coefficient at surface sample i after one diffuse reflection.

The entries in row i of $\mathbf{R}$ are computed by tracing paths sampled over the hemisphere at surface sample i; the delays and attenuations along each path terminating at any other surface sample j are added to the entry $\mathbf{R}_{ij}$ [68]. We can compute a *multi-bounce transfer operator* with n orders of reflection as the matrix sum $\mathbf{T}_n = \mathbf{R} + \mathbf{R}^2 + \cdots + \mathbf{R}^n$. The complete transfer operator $\mathbf{T}$ is given by the limit $\lim_{n\to\infty} \mathbf{T}_n$. However, we truncate the series to a (user-specified) finite number of terms.

4.2.2 Transfer Operator Compression

To apply the transfer operator once, the matrix-vector multiplication in Equation 4.3 needs to be performed once per Fourier coefficient at run-time. However, even for scenes of moderate complexity, the number of surface samples, p, can be very large. Since $\mathbf{R}$ is a $p \times p$ matrix and $\mathbf{l}_n$ is a $p \times 1$ vector, this step takes $O(p^2)$ time per Fourier coefficient per order of reflection, which can quickly become quite expensive. We use the SVD to compute a rank k approximation of $\mathbf{R}$. This allows us to reduce

the complexity to $O(pk)$.

Intuitively, truncating $\mathbf{R}$ to k modes using the SVD removes some of the high spatial frequencies in the transfer operator. A lower-order mode of $\mathbf{R}$ might model reflections from an entire wall, while higher-order modes might model details added to the acoustic response due to local variations in the wall's geometry (such as a painting on the wall). In effect, the parameter k can be used to control the level-of-detail of the acoustic response.

The cost of computing transfer matrices that represent additional bounces can be further reduced to $O(k^2)$ by precomputing appropriate matrices as follows. The direct radiances at each surface sample are stored in the vector $\mathbf{l}_0$. Suppose we have a rank k approximation of $\mathbf{R}$, given by $\widetilde{\mathbf{R}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t$, where $\widetilde{\mathbf{U}}$ is a $p \times k$ matrix, $\widetilde{\mathbf{S}}$ is a $k \times k$ diagonal matrix and $\widetilde{\mathbf{V}}^t$ is a $k \times p$ matrix. Then the first-order radiance at each surface sample is given by:

$$\begin{aligned} \widetilde{\mathbf{R}}\mathbf{l}_0 &= \widetilde{\mathbf{U}}\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t\mathbf{l}_0 \\ &= \widetilde{\mathbf{U}}\mathbf{b}, \end{aligned}$$

where $\mathbf{b} = \widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t\mathbf{l}_0$ is $\mathbf{l}_0$ projected into the span of the first k right singular vectors of $\mathbf{R}$. The second-order response is:

$$\begin{aligned} \widetilde{\mathbf{R}}\widetilde{\mathbf{R}}\mathbf{l}_0 &= \widetilde{\mathbf{U}}(\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t\widetilde{\mathbf{U}})\widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t\mathbf{l}_0 \\ &= \widetilde{\mathbf{U}}\mathbf{D}\mathbf{b}, \end{aligned}$$

where $\mathbf{D} = \widetilde{\mathbf{S}}\widetilde{\mathbf{V}}^t\widetilde{\mathbf{U}}$ is essentially the one-bounce operator in the k-dimensional

subspace spanned by the singular vectors corresponding to the top k singular values of $\mathbf{R}$. The cost of multiplying $\mathbf{b}$ by $\mathbf{D}$ is simply $O(k^2)$. Notice that the third-order response can be written as $\widetilde{\mathbf{U}}\mathbf{D^2}\mathbf{b}$, and so on. This allows us to compute higher-order responses using a $k \times k$ matrix instead of a $p \times p$ matrix.

4.3 Run-time

At run-time, we use the precomputed transfer operator is used as follows:

1. First, we shoot rays from the source to determine the direct radiance at each surface sample.

2. Next, we apply the transfer operator to the direct radiance to obtain the indirect radiance.

3. Finally, we shoot rays from the listener and gather the direct and indirect radiance from each surface sample hit by a ray. These are added to obtain the final echogram at the listener.

Another important design decision is how we compute the transfer operator. There are two broad options:

1. Precompute the SVD approximation of the one-bounce transfer operator. Use the method described in Section 4.2.2 to quickly precompute an approximate multi-bounce operator.

2. Precompute the SVD approximation of the one-bounce transfer operator. At run-time, the orders of reflection can be easily adjusted, perhaps based on compute budget or sound quality.

The first option allows sound designers to rapidly adjust the orders of reflection baked into the precomputed transfer operator. For example, one could first compute a multi-bounce operator with 3 orders of reflection. If the resulting audio at run-time sounds unsatisfactory, the precomputed data can quickly be updated with additional orders of reflection without any further ray tracing or SVD computation.

In the second option, the SVD allows the IR accuracy to be traded off for performance, providing adjustable level-of-detail for sound rendering.

4.4 Results

We now present some experimental results. All tests were performed on an Intel Xeon X5560 workstation with 4 cores (each operating at 2.80 GHz) and 4GB of RAM running Windows Vista. We report timings for all 4 cores since we use Intel MKL to automatically parallelize our matrix operations over all cores of the test machine. We have benchmarked our implementation on three scenes whose complexity is typical of scenes encountered in acoustics applications. Figure 4.2 shows these scenes along with some details.

4.4.1 Performance

For comparison, we chose the state-of-the-art frequency-domain acoustic radiance transfer algorithm [68]. This approach effectively computes the transfer operator (without any SVD approximation) and iteratively applies it to the direct acoustic response until the solution converges. In order to perform a fair comparison, we restrict both ART and our approach to computing 10 orders of reflection.

Table 4.1 summarizes the performance of the precomputation and run-time stages of our algorithm. The run-time complexity depends on the number of modes re-

tained during the SVD approximation; the table clearly highlights this dependency. As shown by the table, our algorithm very efficiently updates IRs when the source position changes at run-time. Note that we precompute a one-bounce transfer operator, and use the approach described in Section 4.2.2 to compute higher-order reflections at run-time. Depending on the application, we could also precompute a multi-bounce operator and apply it directly at run-time, further improving our performance. Our implementation uses a more flexible approach of varying the orders of reflection at runtime. As a result, it is possible to further improve the performance of our implementation.

Figure 4.2: Benchmark scenes. From left to right: (a) Room (252 samples), (b) Hall (177 samples), (c) Sigyn (1024 samples).

Scene	Surface Samples	Precomputation Time **T**	Precomputation Time SVD	Modes	Runtime Initial Scatter	Runtime Transfer Operator	Runtime Final Gather
Room	252	14.2 s	94.5 s	10	43.2 ms	24.0 ms	33.7 ms
				25	45.8 ms	43.8 ms	35.0 ms
				50	42.4 ms	84.3 ms	36.4 ms
Hall	177	13.1 s	93.1 s	10	37.8 ms	26.8 ms	31.5 ms
				25	37.1 ms	45.5 ms	30.2 ms
				50	36.6 ms	79.7 ms	31.2 ms
Sigyn	1024	6.31 min	50.9 min	50	164.1 ms	218.1 ms	109.9 ms

Table 4.1: Performance characteristics of our algorithm. For each scene, we present the precomputation time required by our algorithm for 1K Fourier coefficients. Under precomputation time, we show the time required to compute the transfer operator, **T**, and the time required to compute its SVD approximation. We also compare running times for varying numbers of modes from the SVD. The table shows the time spent at runtime in initial shooting from the source, applying the transfer operator, and gathering the final IR at the listener position.

Table 4.2 shows the benefit of the SVD in compressing the transfer operator.

Scene	Samples	Reference	50 Modes
Hall	177	250.6	161.6
Room	252	508.0	221.6
Sigyn	1024	8388.6	839.2

Table 4.2: Memory requirements of the transfer operators computed by our algorithm with (column 4) and without (column 3) SVD compression. Note that since the entries of each matrix are complex numbers, each entry requires 8 bytes of storage. All sizes in the table are in MB.

The table shows that without using SVD, the transfer operators may be too large to be used on commodity hardware. For the uncompressed ("reference") case, the transfer operator size is $p \times p$, for each Fourier coefficient (1K in our case). For the compressed ("50 Modes") case, the transfer operator size is $p \times k$ for $\tilde{\mathbf{U}}$, $k \times k$ for $\mathbf{D}$ and $k \times p$ for $\tilde{\mathbf{S}}\tilde{\mathbf{V}}^t$, where k is the number of modes retained. In the table, $k = 50$, and p is the number of surface samples in the scene.

Table 4.3 compares the run-time performance of our method and ART. The table shows the time required to update the IRs at the listener when the source moves. The table clearly shows the advantage of our approach. Since our precomputed transfer operator is decoupled from the source position, moving the source does not require recomputing the transfer operator, allowing the source position to be updated much faster than would be possible with ART.

Table 4.3 can also be used to derive the performance of our algorithm for the case when a multi-bounce transfer operator is precomputed. For example, suppose we precompute a transfer operator with 10 orders of reflection for the Sigyn scene. Then the run-time cost would be the same as that of the one-bounce operator, i.e., 468.5 ms. The difference, i.e. $(512.8ms - 468.5ms) \times 1024 = 45.4s$ would be the additional time spent during preprocessing to derive the multi-bounce operator from the one-bounce operator (the factor of 1024 arises due to the fact that the timings in Table

Scene	Orders	Radiance Transfer	Direct-to-Indirect Transfer (50 modes)
Room	2	11.7 s	131.8 ms
	5	11.8 s	154.4 ms
	10	12.0 s	179.3 ms
Hall	2	10.6 s	116.5 ms
	5	10.7 s	147.3 ms
	10	10.9 s	170.5 ms
Sigyn	2	185.3 s	468.5 ms
	5	186.7 s	491.2 ms
	10	188.7 s	512.8 ms

Table 4.3: Comparison of our approach with ART. We compare the time required by our algorithm to update the source position and recompute the IR at the listener position after a varying number of diffuse reflections. Since ART does not decouple the transfer operator from the source position, it needs to perform a costly recomputation of the transfer operator each time the source moves. On the other hand, our algorithm quickly updates the direct IR at all surface samples, then applies the precomputed transfer operator. This allows our approach to handle moving sources far more efficiently than ART.

4.3 are for matrix-vector multiplication, whereas precomputing the multi-bounce operator from the one-bounce operator requires matrix-matrix multiplications).

4.4.2 Analysis

Figure 4.3 compares the output of our algorithm and ART. The figure shows energy decay curves, smoothed using a moving-average low-pass filter, for different numbers of modes. As the figure shows, reducing the number of modes significantly (down to 50 modes) has very little effect; however, if far fewer modes are used, significant errors appear in the energy decays, as expected. Coupled with the memory savings demonstrated in Table 4.2 and performance advantage demonstrated in Table 4.3, we see that using the SVD allows us to significantly reduce memory requirements and increase performance without significant degradation of the computed IRs. Along

with the plots, Figure 4.3 shows RT_{60} (reverberation time) values estimated from the decay curves. The data demonstrates that SVD approximation upto 50 modes does not lead to significant change in reverberation time. Of course, the best way to demonstrate the benefit of our approach is by comparing audio clips; for this we refer the reader to the accompanying video [2].

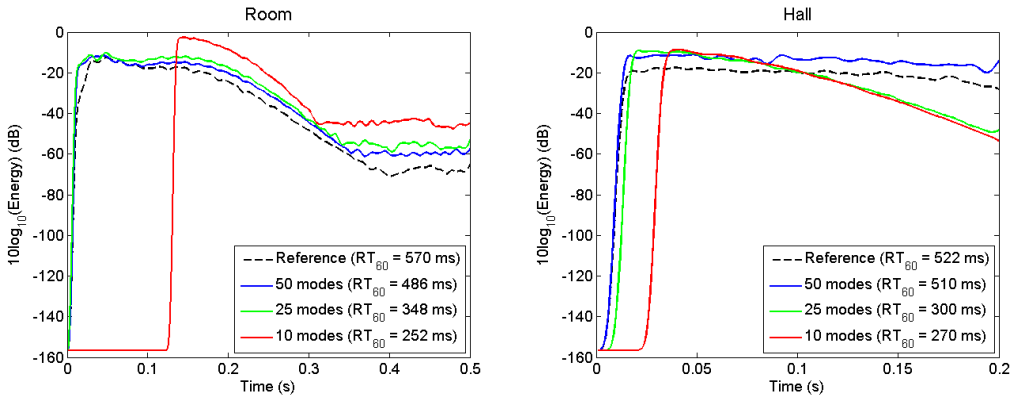


Figure 4.3: Comparison of diffuse IRs (30 orders of reflection, absorption coefficient 0.2) computed by our system with and without SVD compression, for some of our benchmark scenes. The plots show squared IRs, smoothed using a moving-average low-pass filter.

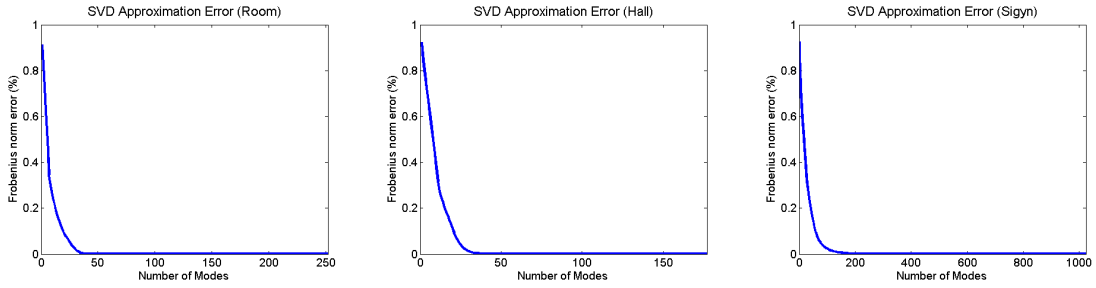


Figure 4.4: SVD approximation error for transfer operators. For each benchmark scene, the plots show the relative Frobenius norm error of rank-k approximations of $\mathbf{T}$ (for one value of ω) for all possible values of k. From left to right: (a) Room (252 samples), (b) Hall (177 samples), (c) Sigyn (1024 samples).

The SVD approximation error of the transfer operator is measured using the Frobenius norm. Figure 4.4 plots the error against the number of modes retained

[2] `http://gamma.cs.unc.edu/Sound/diffuse/`

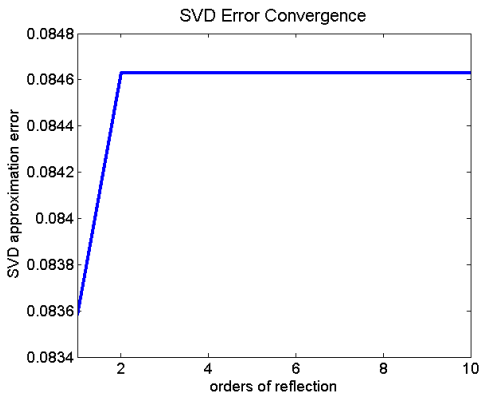


Figure 4.5: SVD approximation error for each higher order of reflection, for the Sigyn scene (see Figure 4.2).

in the transfer operator. The figure suggests that we could potentially use a very small number of modes to compute IRs with diffuse reflections at runtime. Figure 4.5 plots the SVD approximation error (at 50 modes) with increasing orders of reflection. The figure clearly shows that the error introduced by the SVD approximation for higher orders of reflection quickly converges. In other words, the IR energy due to higher-order reflections can be modeled using very few SVD modes of the transfer operator. This matches the intuition that higher-order reflections have low spatial frequency. As a result, when computing very high orders of reflection (say 50), we can use very few SVD modes beyond the first 2-3 orders while still capturing the higher order energy (which must be captured to model the late reverberation tail of the IR) accurately.

Chapter 5

Compact Acoustic Transfer Operators

In this chapter, we discuss another specific algorithm based on the PART framework [6]. The algorithm performs its calculations in the time domain. The development of this algorithm is motivated by two goals. First, we wish to incorporate specular reflections and diffraction in the transfer operator. Second, since the sizes of the frequency-domain transfer operators computed using the algorithm of Chapter 3.3.3 are still quite large (several hundred megabytes), we wish to develop more compact representations of the transfer operator. The algorithm presented in this chapter generates compact transfer operators using the Karhunen-Loeve transform, and uses multi-bounce ray tracing and a two-pass run-time algorithm to simulate specular reflections as well as diffraction.

5.1 Precomputation

We assume that acoustic radiance at a surface sample does not vary with direction (i.e., the surface samples are diffuse emitters and receivers). In other words, the transfer operator models sound energy which is emitted uniformly in all directions from a given surface sample, and propagates through the scene (undergoing several diffuse and specular reflections as well as diffraction) until the propagation is finally termi-

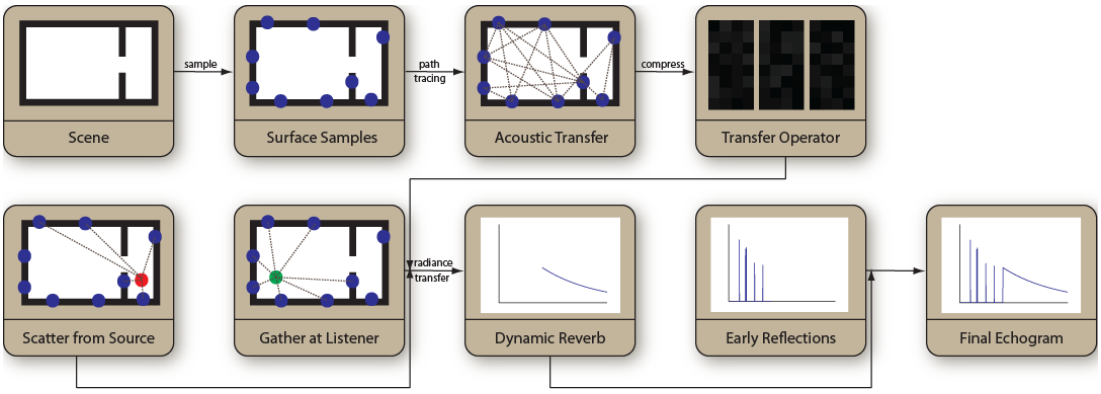


Figure 5.1: Overview of our algorithm. **Top row**: Precomputation. **Bottom row**: Run-time interactive sound propagation.

nated upon incidence at some other surface sample. The propagated, incident sound field is averaged over all incidence directions, resulting in a directionally-invariant indirect acoustic radiance at each surface sample. This simplifying assumption is motivated by the fact that after a few orders of reflection, most of the sound energy in a scene would have typically undergone diffuse reflections [39]. This may result in some higher-order echoes being replaced with reverberation, but can be corrected when computing the early response. We now describe our approach for computing a compact representation of the acoustic transfer operator.

5.1.1 Transfer Operator Precomputation

In order to define the acoustic transfer operator for the scene, we first sample n points on the surface of the scene using area-weighted sampling [31] (Figure 5.1 (b)). We then construct a compact, scene-dependent KLT basis for representing echograms (Figure 5.1 (c)), which we then use to compress echograms computed between each surface sample (Figure 5.1 (d)).

We use energy-based path tracing (i.e., Monte Carlo integration of the acoustic rendering equation) to compute the sample-to-sample echograms. When each path

encounters a geometric primitive, it can be diffusely reflected, specularly reflected or diffracted, depending on material properties. Attenuations are applied according to standard geometric acoustics models as discussed below.

Diffuse Reflections Rays are diffusely reflected as per the Lambertian model by randomly sampling a direction on the hemisphere at the point of incidence, and sending a reflected ray along the sampled direction. The ray's energy is attenuated by the frequency-dependent *diffuse coefficient* $d(\nu) = (1 - \alpha(\nu))\sigma(\nu)$, where $\alpha(\nu)$ is the frequency-dependent absorption coefficient and $\sigma(\nu)$ is the frequency-dependent scattering coefficient of the surface material.

Specular Reflections Specular reflection of rays is performed by reflecting incident rays as per the laws of reflection. The ray's energy is attenuated by the frequency-dependent *specular coefficient* $s(\nu) = (1 - \alpha(\nu))(1 - \sigma(\nu))$.

Edge Diffraction Diffraction is modeled using an energy-based ray tracing model derived from Heisenberg's uncertainty principle [74, 73]. Rays which pass sufficiently close to a diffracting edge [74] are diffracted by deviating them in the plane normal to the diffracting edge. The angle of deviation is randomly sampled from a frequency-dependent probability distribution.

5.1.2 Echogram Representation

In order to capture closely-spaced echoes, which may arise in 2^{nd} or 3^{rd} order reflections captured in the transfer operator, we sample echograms at the audio sampling rate of 48 kHz. As a result, it is impractical to store precomputed sample-to-sample echograms in the time domain, since this would require 192 kB per second per

echogram. For $n \approx 256$ surface samples, this would result in the transfer operator requiring 12 GB of storage per second.

Frequency-domain representations have been used in prior precomputation-based sound propagation algorithms, but require a very large number of coefficients ($m \approx 1024$) to represent either the echograms themselves [68], or the decay envelopes of the echograms [72] (which cannot be used to model sharp echoes arising from 2^{nd} or 3^{rd} order reflections).

Commonly used signal compression techniques are based on representing the signals using transforms such as the Fourier transform, the discrete cosine transform (DCT) and the related modified discrete cosine transform (MDCT) [92], and wavelet representations. Fourier and DCT representations require a few thousand coefficients [72, 68] in order to represent the wide range of audible sound frequencies. While the MDCT and wavelet transforms are typically sparse, they too require hundreds of coefficients in order to represent middle-to-high-frequency reverberation in large spaces. Ideally, we would prefer a basis in which echograms can be represented using relatively few coefficients.

For this, we use a scene-dependent Karhunen-Loeve basis, derived using the Karhunen-Loeve Transform (KLT) [49]. The KLT is defined as follows. In order to derive an orthogonal basis for a d-dimensional vector space $\mathbf{S}$, we first randomly sample some number (say p) of vectors in the space. These vectors are written as column vectors and placed side-by-side to form the *data matrix* $\mathbf{A}_{d\times p}$ (subscripts denote matrix dimensions). We can then use the singular value decomposition (SVD) to decompose the data matrix: $\mathbf{A}_{d\times p} = \mathbf{U}_{d\times p}\mathbf{\Sigma}_{p\times p}\mathbf{V}^t_{p\times p}$. The columns of the orthogonal matrix $\mathbf{U}$ are then used as a basis set for $\mathbf{S}$.

To generate an orthogonal basis for sample-to-sample echograms in a given scene, we first randomly choose p pairs of surface samples, and compute echograms between

them (using path tracing). The dimension of the vector space in which all echograms lie is equal to the number of samples used to represent the echograms in the time domain. These echograms are used to form the data matrix, and then the SVD is used to compute the KLT basis matrix $\mathbf{U}$. Since the basis vectors are sorted in decreasing order of singular values, we can truncate $\mathbf{U}$ and retain only the first m columns. As demonstrated in the accompanying video, the approximation error can be barely perceptible (in our benchmarks), even with very few basis vectors ($m \approx 32 - 64$).

In essence, this formulation "learns" a good basis for representing echograms in a given scene by using several *example echograms* computed in the scene. Assuming the surface sample pairs used to generate the example echograms are distributed throughout the scene, the Karhunen-Loeve transform can be used to estimate a basis of echograms that requires the fewest number of coefficients to represent an echogram in the scene for a given approximation error. Furthermore, since the storage and time complexity of this algorithm scales linearly with m, we choose the Karhunen-Loeve basis to represent the acoustic transfer operators compactly.

5.2 Run-time

At run-time, we use an approach similar to prior visual rendering algorithms [91] and compute sound propagation effects using a two-pass algorithm (see Figure 5.1). The two passes work as follows:

1. **Early Response using Ray Tracing.** Since low-order specular reflections and diffraction are important for sound localization, low-order reflections (diffuse and specular) and edge diffractions are computed using path tracing [11].

2. **Late Response using Radiance Transfer.** We analytically compute the di-

rect echogram at each surface sample due to the (potentially moving) source(s) (Figure 5.1 (f)). The acoustic transfer operator is then applied to the direct echograms; this yields echograms at each surface sample which model higher-order reflections and diffraction. The resulting echograms are gathered from the surface samples at the listener (Figure 5.1 (g)) to quickly compute the higher-order echogram from a moving source to a moving listener (Figure 5.1 (h)).

We now detail each pass of our algorithm.

5.2.1 Acoustic Radiance Transfer

The direct echogram due to a single source at surface sample j can be completely characterized by a delayed impulse with (distance) attenuation a_j^s and a delay d_j^s. Similarly, the response at a listener due to direct sound along each gather ray i can be completely characterized by a delayed impulse with (distance) attenuation a_i^l and a delay d_i^l.

For simplicity, the BRDFs at the first and last reflections are multiplied into the acoustic transfer operator. Furthermore, for simplicity of exposition, we assume that the number of gather rays traced from the listener is also n; in practice, we trace $O(n)$ gather rays, with the constant factor chosen based on run-time performance. As each gather ray hits a point on the surface of the scene, the point is mapped to a surface sample using nearest-neighbor interpolation. We denote the surface sample corresponding to gather ray i by $S(i)$.

These attenuations and delays are then combined with the compressed acoustic transfer operator to compute the final echogram as follows. We denote the precomputed echogram from sample j to sample $S(i)$ by $L_{i,j}(t)$. Then the energy received at the listener via propagation paths whose first reflection occurs at sample j and

last reflection occurs at sample $S(i)$ is given by:

$$E_{i,j}(t) = a_j^s a_i^l L_{i,j}(t - d_j^s - d_i^l), \tag{5.1}$$

and the final echogram at the listener is obtained by adding together energy received from all possible propagation paths:

$$E(t) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_j^s a_i^l L_{i,j}(t - d_j^s - d_i^l). \tag{5.2}$$

Since the sample-to-sample echograms in the transfer operator are stored in a basis with m coefficients, we use the basis expansion to obtain:

$$L_{i,j}(t) = \sum_{k=1}^{m} \alpha_{i,j}^k b^k(t), \tag{5.3}$$

$$E(t) = \sum_{k=1}^{m} \Big(\sum_{i=1}^{n} \sum_{j=1}^{n} a_j^s a_i^l \alpha_{i,j}^k \Big) b^k(t - d_j^s - d_i^l), \tag{5.4}$$

where b^k denotes the k^{th} basis function and the α's are coefficients of echograms in the basis space. The above expression can be reformulated as a sum of convolutions:

$$E(t) = \sum_{k=1}^{m} H^k(t) \otimes b^k(t), \tag{5.5}$$

$$H^k(t) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_j^s a_i^l \alpha_{i,j}^k \delta(t - d_j^s - d_i^l). \tag{5.6}$$

Therefore, at run-time, we use the source position to quickly update a_j^s and d_j^s; and the listener position to quickly update a_i^l and d_i^l. These are used along with the compressed transfer operator to construct the convolution filters $H^k(t)$; convolving

the echogram basis functions with these filters and accumulating the results yields an echogram representing higher-order reflections and diffraction from the source to the listener.

Low-Order Effects Since we assume that surface samples are diffuse emitters and receivers, the radiance transfer pass cannot model all kinds of propagation paths. Consider a variant of Heckbert's regular expression notation [32] for propagation paths, with D denoting a diffuse reflection, S denoting a specular reflection, and E denoting an edge diffraction. Then the radiance transfer pass is restricted to computing $D(D|S|E)^*D$ paths.

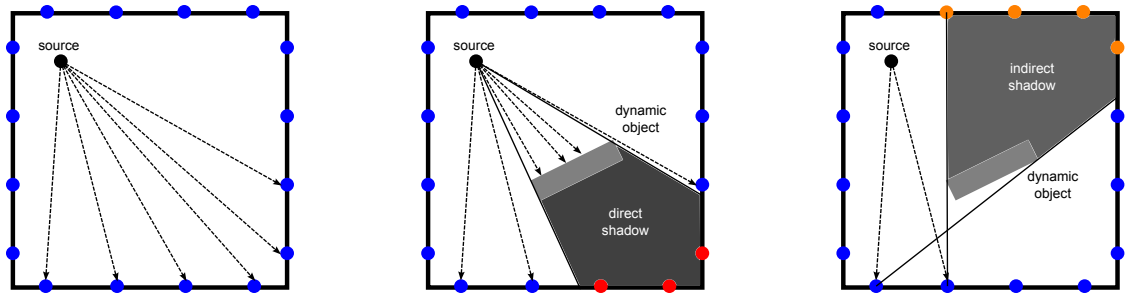
However, low-order specular reflections and diffraction provide important directional cues to the listener ([38], pp. 194). Therefore, in the first pass of our algorithm, low-order path tracing is performed to compute 1-3 orders of specular reflections and edge diffraction as well as first-order diffuse reflections. At each specular reflection or edge diffraction, energy can be converted to diffuse energy and transferred to the precomputed transfer operator, allowing paths of the form $(S|E)^q(D|S|E)^*(S|E)^q$ (for low values of q) to be modeled, thus allowing low-order purely specular and purely diffraction paths to be modeled. As can be seen from the corresponding regular expressions, the paths modeled in the first and second passes are disjoint (i.e., no path is traced in both passes), hence the echograms from each pass can be directly added, along with the direct source-to-listener contribution, to determine the final echogram at the listener.

5.2.2 Dynamic Scenes

The acoustic transfer operator is inherently decoupled from both source and listener positions. As a consequence of this formulation, our algorithm can compute higher-order sound propagation in scenes with moving sources and listeners, as mentioned above. Moreover, the computation of early reflections is performed using ray tracing, and hence we can handle fully dynamic scenes in the ER pass.

Dynamic objects may also affect the late response. We use interactive ray tracing [80] to compute direct echograms at each surface sample (Figure 5.2 (a)). As a result, these rays can intersect and be blocked by dynamic objects (Figure 5.2 (b)). This allows dynamic objects to induce a "shadow" region and reduce the energy in the direct echograms at the surface samples in the shadow region (see Figure 5.2 (b)). Since these (modified) direct echograms are used as input to the precomputed acoustic transfer operator in the first pass, our formulation allows dynamic objects to modify (to a limited extent) the propagated sound field heard at the listener in the LR pass. Similarly, since interactive ray tracing is used in the final gather step, reflected and/or diffracted sound can be occluded by a dynamic object before it reaches the listener.

However, since the transfer operator is pre-computed for surface samples defined over static geometry only, we cannot model reflections or inter-reflections off dynamic objects in the radiance transfer pass. For example, we can only model ER (and not LR) due to sound reflecting off a moving car. Furthermore, since the transfer operator is computed over static surfaces only, we cannot model "indirect shadow" regions – i.e., occlusion of reflected days by dynamic objects (Figure 5.2 (c)). For example, we cannot accurately model the case where two static rooms are separated by a dynamic door, since the precomputed transfer operator cannot take into account the changes



(a) (b) (c)

Figure 5.2: Dynamic source shadowing. (a) A sound source in a static room. Blue dots indicate surface samples. (b) Adding a dynamic object (in this case, a rectangle). Some rays traced from the source are blocked by the dynamic object. This induces a "shadow" region and changes the direct response at the red dots. This in turn would affect the indirect response everywhere. This effect is modeled by our algorithm. (c) Direct energy reflected from surface samples may also be occluded by the dynamic object, inducing an "indirect shadow" region. However, since we precompute sample-to-sample transfer, indirect shadows are not modeled by our algorithm.

in visibility between surface samples on the walls of the two rooms caused by opening or closing the door.

5.2.3 Run-time Error Control

One of the advantages of our choice of echogram basis is that it allows run-time control over the accuracy of the sound propagation. Using fewer basis coefficients at run-time allows accuracy to be adapted to a limited compute budget without sacrificing the frequency content of the propagated sound. The SVD used to compute the Karhunen-Loeve basis for a scene implicitly sorts the basis functions such that most of the energy is distributed into the first few basis functions (see Figure 5.4). The remaining basis functions can be ignored at run-time by truncating the SVD, with only a minor impact on the accuracy of the echograms computed. Since the run-time performance scales linearly with the number of basis coefficients used (see Section 5.3.2), we can increase performance by using fewer basis coefficients, at the

Figure 5.3: Benchmark scenes for compact acoustic transfer operators. From left to right: Sibenik (80K triangles), Movie Theater (120K triangles), Basement (0.5K triangles), and Attic (1K triangles).

cost of a slight reduction in sound quality (as shown in the accompanying video).

5.3 Results

We now present details of our implementation and experimental results demonstrating its performance. Our implementation is written in C++, compiled using Microsoft Visual Studio 2010. We use Intel Math Kernel Library (MKL) for parallelization of linear algebra operations, and NVIDIA OptiX for interactive ray tracing. All precomputation and run-time tests were performed on an Intel Core 2 Quad 2.83 GHz CPU with 4GB RAM and an NVIDIA GeForce GTX 480 GPU, running Windows 7. Figure 5.3 shows the benchmark scenes used in our experiments.

Detecting Diffraction Paths We exploit the flexible nature of OptiX ray tracing kernels [57] to allow the ray tracer to detect rays passing close to diffracting edges. Diffracting edges are first detected based on the relative orientations of their incident triangles [79]. For each diffracting edge, a bounding box is added to the OptiX scene graph, using a *bounding box program*. The thickness of these bounding boxes is user-specified. During ray tracing, rays that intersect the edge bounding boxes are detected using an intersection program. These rays recursively spawn new diffraction rays using a closest hit program. An energy-based formulation is used to derive the attenuation due to diffraction. In this manner, separate kernels (a bounding box

program, intersection program, and a closest hit program) are used in conjunction to detect diffraction paths for rays that pass close to diffracting edges.

5.3.1 Performance

Table 5.1 shows the performance of the precomputation phase of our algorithm, as well as the storage requirements of the precomputed acoustic transfer operators. For each scene, we show the time spent in computing example echograms (column 4) and using them to construct a basis for echograms (column 5). We also show the time required to precompute the compressed acoustic transfer operator for each scene (columns 7 and 8). m refers to the number of example echograms used for basis construction, and n refers to the number of surface samples chosen over the surface of the scene. Finally, we show the storage required for the echogram basis in column 6, and for the transfer operators in column 9. As the table shows, our algorithm can compute compact acoustic transfer operators which require only a few tens of megabytes of storage within a few tens of minutes.

Table 5.2 demonstrates the performance of our two-pass run-time algorithm. For each scene, we show the time spent in each stage of the run-time algorithm. Column 5 shows the time taken to compute direct echograms from the source at each surface sample. Column 6 shows the time required to apply the transfer operator. Column 7 shows the time required to gather the higher-order echograms from each surface sample. Column 8 shows the time required to compute the early response using ray tracing. The table shows that our algorithm can efficiently compute higher-order reflections of sound, even for complex models consisting of tens or hundreds of thousands of triangles. Note that two of the scenes (Basement and Attic) are not shown in Table 5.2. This is because these scenes are rendered within the game engine, so the corresponding performance numbers are not representative of our

Scene	Triangles	Echogram Basis				Transfer Operator		
		m	Prop. (s)	Constr. (s)	Size (MB)	n	Computation (s)	Size (MB)
Sibenik	80K	256	130.29	3.34	46.9	128	481.87	16.0
Movie Theater	120K	256	186.45	0.75	46.9	256	2243.17	64.0
Attic	1128	64	27.81	0.09	11.7	64	105.66	1.0
Basement	548	64	23.52	0.07	11.7	64	93.66	1.0

Table 5.1: Performance and memory overhead of our precomputation algorithm.

Scene	Triangles	m	n	Scatter	Transfer	Gather	ER	**Total**	FPS
Sibenik	80K	32	128	0.4	149	42.5	1.4	193.3	5.2
Movie Theater	120K	16	256	0.6	77	82.8	2.1	162.5	6.1

Table 5.2: Performance of our run-time implementation. All times are in milliseconds.

stand-alone OptiX-based implementation. In particular, the game engine's ray tracer is not optimized for ray-traced rendering workloads. As the accompanying video [1] demonstrates, we still obtain sound propagation update rates of 5-10 FPS within the game engine.

5.3.2 Time and Storage Complexity

During precomputation, path tracing is performed from each of the n surface samples, to determine echograms between each pair of surface samples. These n^2 echograms are then compressed into the Karhunen-Loeve basis with m coefficients. Hence, storing the compressed acoustic transfer operator requires $O(mn^2)$ memory. Projecting each echogram into the basis using a matrix-vector product requires $O(mT)$ time, where T is the number of time-domain samples used to represent the uncompressed echogram. Therefore, the total time required to compress the acoustic transfer operator is $O(mn^2T)$.

At run-time, the scatter and gather steps involve $O(n)$ work each; computing each convolution filter $H^k(t)$ takes $O(n^2)$ time to evaluate the double summation in

[1] `http://gamma.cs.unc.edu/CATR/`

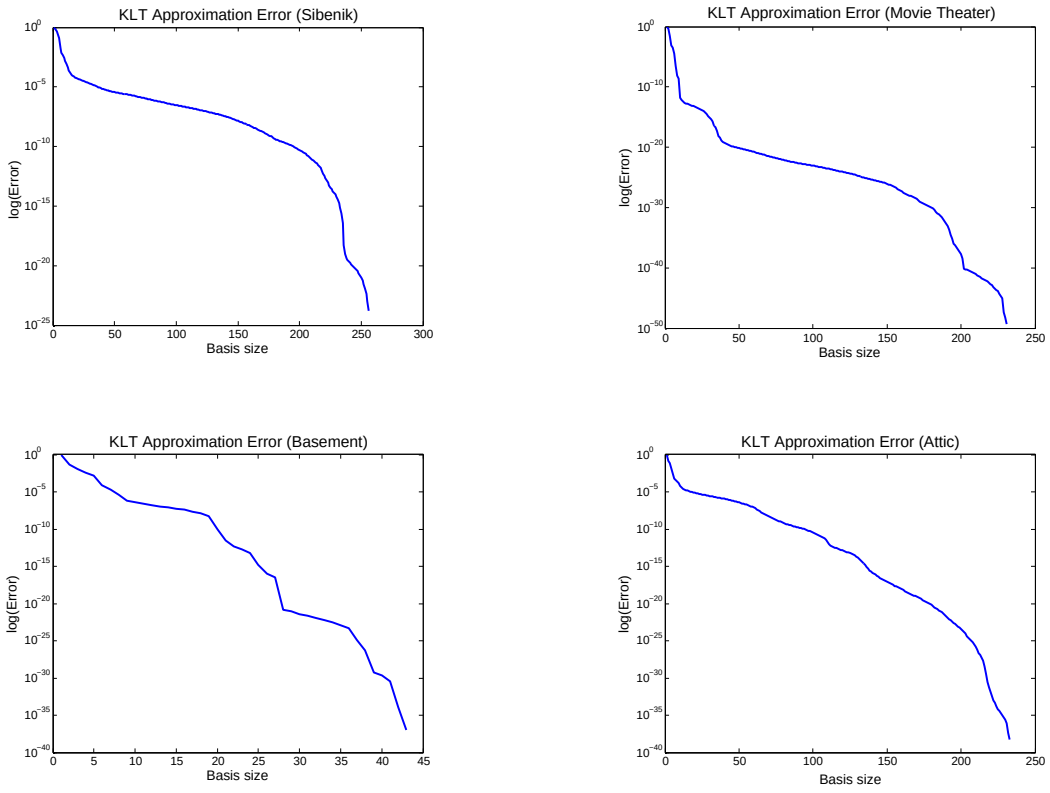


Figure 5.4: Relative Frobenius norm error due to SVD truncation during KLT basis construction, for different scenes.

Equation 5.6. The total time required to compute the convolution filters is hence $O(mn^2)$. The basis functions $b^k(t)$ are stored in the frequency domain, hence we can use the Fourier theorem to quickly compute the convolutions in Equation 5.6 in $O(mT \lg T)$ time.

5.3.3 Choice of Parameters

There are several parameters that need to be appropriately chosen when using our algorithm to compute and use acoustic transfer operators: s, the number of samples in an echogram; n, the number of surface samples; p, the number of example echograms used for basis construction; and m, the number of basis functions retained

at run-time. We now discuss our choice of values for these parameters as used in our experiments.

Echogram Length The echogram length can be determined using the expected reverberation time of the scene. If the length of the echograms is chosen to be T seconds, then at our sampling rate of 48 kHz, the size of each basis function is $s = 48000T$ samples. In our experiments, we use values of T ranging from 1–2 seconds.

Surface Samples The number of surface samples to generate for each scene can be determined experimentally, guided by the fact that it is not possible to distinguish directions of incidence of sound with as much resolution as it is possible to distinguish directions of incidence of light [82]. Audio clips generated with varying numbers of surface samples can be found in the accompanying video.

Basis Generation The values of p, i.e., the number of example echograms to use for basis construction, were arrived at through experiment. We used values of $p \in [64, 512]$ to generate the KLT basis. We then used plots of Frobenius norm error computed for the data matrix $\mathbf{A}$ to determine a sufficient value of p. Some resulting plots of Frobenius norm error are shown in Figure 5.4. Note that we randomly chose the surface sample pairs to generate the example echograms. For more complex environments with multiple connected rooms with a large amount of occlusion, it would be necessary to ensure (at least) that example echograms are computed between each pair of adjacent rooms.

These plots were also used to determine sufficient values for m, i.e., the number of basis functions used at run-time. As the plots show, low values of m ($\approx 32 - 64$) can be used without significant Frobenius norm error. Figure 5.5 shows energy decay

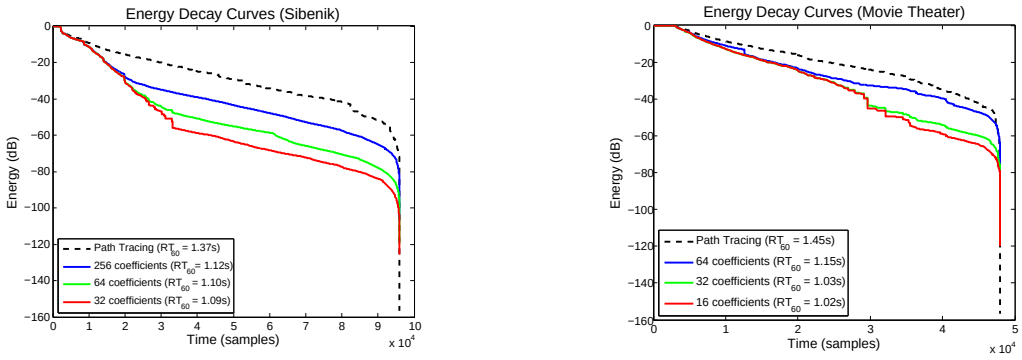


Figure 5.5: Energy Decay Curves for different scenes, with varying numbers of KLT coefficients.

curves computed for varying values of m, compared with energy decay curves for reference path tracing solutions. The plots show that m provides a straightforward way to increase accuracy (at the cost of performance). Audio clips generated with varying numbers of KLT basis functions can be found in the accompanying video. These clips also show that low values of m can be used at run-time without significant degradation of audio quality.

Chapter 6

Ambient Reverberance

In Chapters 3–5, we presented Precomputed Acoustic Radiance Transfer, an approach for efficiently simulating higher-order sound propagation effects in interactive applications. However, the algorithms still required a few hundred milliseconds to update impulse responses, and required tens to hundreds of megabytes to store their precomputed data. This may make them too costly for applications intended to run on commodity hardware.

Therefore, in this chapter, we describe Ambient Reverberance [9], an algorithm for computing spatially- and directionally-varying reverberation in complex, dynamic scenes. The algorithm uses the local geometry around the listener to dynamically update the parameters of a statistical reverberation filter. It is able to compute reverberation effects in a few milliseconds, without requiring any precomputed data.

6.1 Artificial Reverberation

In large, reverberant scenes, using wave-based or geometric simulation for computing an impulse response containing the reverberation effects is usually impractical.

Therefore, reverberation is modeled using *artificial reverberators*, which are essentially implementations of *infinite impulse responses* (IIRs) such as the following:

$$y(t) = \sum_{i=1}^{N} c_i s_i(t) + dx(t), \tag{6.1}$$

$$s_i(t + \Delta t_i) = \sum_{j=1}^{N} a_{i,j} s_j(t) + b_i x(t). \tag{6.2}$$

where $x(t)$ is the input signal, $y(t)$ is the output signal (with reverberation effects added), and $a_{i,j}$, b_i, c_i, and d are constants. The values of these constants are, in turn, determined by the values of the parameters of the statistical reverberation model being used.

6.2 Reverberation Time

One of the most important parameters of any statistical reverberation model is the *reverberation time*, denoted by RT_{60}, which is defined as the time required for sound energy to decay by 60 dB, i.e., to one millionth of its original strength, at which point it is considered to be inaudible [23].

One of the earliest statistical methods for determining RT_{60} was the Sabine equation, originally developed for single rooms based on empirical observations:

$$RT_{60} = \frac{k}{c} \frac{V}{S\alpha}, \tag{6.3}$$

where V is the volume of the room, S is its surface area, α is the average absorption coefficient of the surfaces in the room, c is the speed of sound, and k is a constant of proportionality. However, due to the presence of the α factor in the

denominator, this model does not work well for absorption coefficients below around 0.3. The *Eyring model* addresses this issue by modifying the absorption term in the denominator:

$$RT_{60} = \frac{k}{c}\frac{V}{S\log(1-\alpha)}. \tag{6.4}$$

In either case, the statistical model essentially describes reverberation as an exponential decay:

$$\begin{aligned} E(t) &= E_0 e^{k\frac{t}{RT_{60}}} && (6.5)\\ &= E_0 e^{\frac{cS}{4V}t\log(1-\alpha)}, && (6.6) \end{aligned}$$

where E_0 is a constant. The exponential decay is typically modulated by a noise function to provide more variation in the output audio.

6.3 Mean Free Path

Intuitively, the reverberation time is related to the manner in which sound undergoes repeated reflections off of the surfaces in the scene. This in turn is quantified using the *mean free path* μ, which is the average distance that a sound ray travels between successive reflections.

In other words, consider tracing rays with random directions and with origins at random points in the environment. Every time a ray intersects a surface, it is reflected in a random direction (up to a maximum number of reflections). The distance between two reflections is then averaged over all reflections and over all paths, and the resulting value is called the mean free path.

Mathematically, the mean free path and the reverberation time are related as follows [38]:

$$T = k\frac{\mu}{\log(1-\alpha)}, \tag{6.7}$$

where T is the reverberation time, μ is the mean free path, α is the average surface absorption coefficient, and k is a constant of proportionality. Note that for a single rectangular room, $\mu = \frac{cS}{4V}$, and it can be shown that Equation 6.7 can be reduced to the Eyring model.

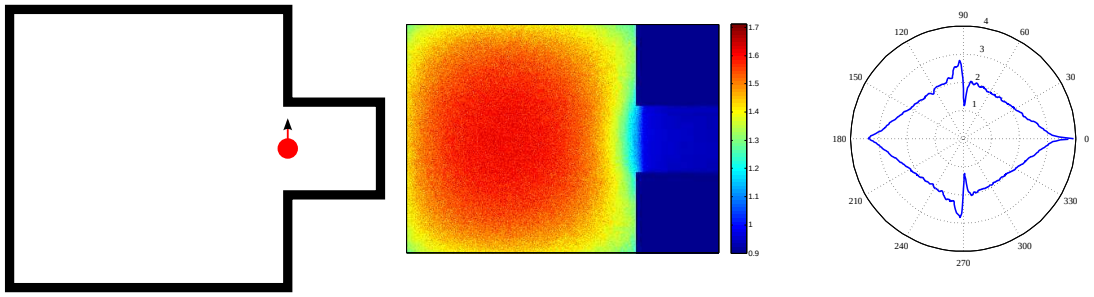
Next, we describe an approach for adjusting a user-controlled mean free path based on local geometry information.

6.4 Spatially-Varying Reverberation

The mean free path varies with listener position in the scene, as shown in Figure 6.1. A straightforward approach for computing the mean free path would be to use path tracing to sample a large number of multi-bounce paths, and compute the mean free path from first principles. However, like ambient occlusion, we only use local visibility and depth information. We define a function $l(\omega)$, which denotes the distance from the listener to the nearest surface along direction ω. We integrate over a unit sphere centered at the listener's position to determine the *local distance average*, $\bar{l}$:

$$\bar{l} = \frac{1}{4\pi}\int l(\omega)d\omega. \tag{6.8}$$

Figure 6.2 illustrates this process. This approach is similar in spirit to the process of integrating a visibility function when computing ambient occlusion. The above integral is evaluated using Monte Carlo integration. We trace rays out from the



(a) A coupled-room scene. (b) Spatial variation of mean free path. (c) Directional variation of mean free path

Figure 6.1: Spatial and directional variation of mean free path. **Left**: A 3m×3m×1m room adjacent to a 1m×1m×1m room. **Center**: Variation of mean free path over the two-room scene, with varying listener position. Colors indicate mean free path in meters. Note the smooth transition between mean free paths (and hence, between reverberation times) at the doorway connecting the two rooms. **Right**: Variation of mean free path with direction of incidence at the listener position indicated by the red dot, with the listener's orientation indicated by the arrow. The difference between the left and right lobes, due to the different sizes of the rooms on either side, indicates that more reverberant sound should be received from the left than from the right.

listener, and average the distance travelled by each ray, denoting the result by $\bar{l}$. A reference reverberation time T_0 is specified for the scene; we use this to determine a reference mean free path μ_0 as per Equation 6.7.

We then blend the user-controlled mean free path μ_0 and the local distance average $\bar{l}$:

$$\mu = \beta\bar{l} + (1 - \beta)\mu_0, \tag{6.9}$$

where $\beta \in [0, 1]$ is the local blending weight, and μ is the adjusted mean free path. While β may be directly specified to exaggerate or downplay the spatial variation of reverberation, we describe a systematic approach for determining β based on surface absorption.

Suppose reverberated sound undergoes n reflections before bouncing to the lis-

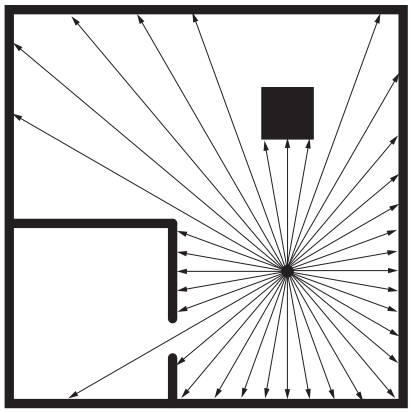


Figure 6.2: Sampling directions around a listener to determine a local distance average. In this top-down view, solid black denotes a solid surface. The arrows denote rays traced to sample distance from a point listener at the (common) origin of the rays.

tener. Therefore, the distance traveled before the final bounce is (on average) $n\mu_0$, and the total distance traveled upon reaching the listener is (on average) $\bar{l} + n\mu_0$. Averaging over all $n + 1$ bounces yields:

$$\mu = \frac{1}{n+1}\bar{l} + \frac{n}{n+1}\mu_0, \tag{6.10}$$

$$\beta = \frac{1}{n+1}. \tag{6.11}$$

Intuitively, the linear combination of Equation 6.9 serves to update an average – the mean free path – with the data given by the local distance average. As per the definition of RT_{60} [38], sound energy decays by 60 dB after undergoing n bounces. Each bounce reduces sound energy by a factor of α. Therefore:

$$(1-\alpha)^n = 10^{-6}, \tag{6.12}$$

$$n = \frac{-6\log 10}{\log(1-\alpha)}, \tag{6.13}$$

The above expressions allow the reverberation time to be efficiently adjusted as a function of the local distance average and surface absorption properties.

6.5 Directionally-Varying Reverberation

Mean free paths also vary with direction of incidence, as shown in Figure 6.1. The above technique can be easily generalized to obtain direction-dependent reverberation times from a *single* user-controlled reverberation time. We express μ as a function of incidence direction ω:

$$\mu(\omega) = \beta l(\omega) + (1 - \beta)\mu_0. \tag{6.14}$$

Here $\mu(\omega)$ denotes the average distance that a ray incident at the listener along direction ω travels between successive bounces. As before, $l(\omega)$ is computed using Monte Carlo sampling from the listener position. Note that here, ω refers to the direction of incidence at the listener, after any and all reflection or scattering. We then use a spherical harmonics representation of l to obtain directional reverberation, since spherical harmonics are well-suited for representing smoothly-varying functions of direction.

Spherical Harmonics Spherical harmonics (SH) are a set of basis functions used for representing functions defined over the unit sphere. SH bases are widely used in computer graphics to model the directional distribution of radiance [71]. The basis functions are defined as [70]:

$$Y_{p,q}(\theta,\phi) = N_{p,q}\, e^{iq\phi}\, P_{p,|q|}(\cos\theta), \tag{6.15}$$

$$N_{p,q} = \sqrt{\frac{(2p+1)(p-|q|)!}{4\pi(p+|q|)!}}, \tag{6.16}$$

where $p \in \mathbf{N}$, $-p \leq q \leq p$, $P_{p,q}$ are the associated Legendre polynomials, and $\omega = (\theta, \phi)$ are the elevation and azimuth, respectively. Here, p is the *order* of the SH basis function, and represents the amount of detail captured in the directional variation of a function. Guided by the above definitions, we project $l(\omega)$ into a spherical harmonics basis:

$$l(\omega) = \sum_{p=0}^{P} \sum_{q=-p}^{p} l_{p,q} Y_{p,q}(\omega), \tag{6.17}$$

$$\mu(\omega) = \sum_{p=0}^{P} \sum_{q=-p}^{p} \mu_{p,q} Y_{p,q}(\omega). \tag{6.18}$$

The linearity of spherical harmonics allows us to independently adjust the SH coefficients of the mean free path:

$$\mu_{p,q} = \beta l_{p,q} + (1-\beta)\mu_0. \tag{6.19}$$

Multi-channel Reverberation Modern video games and VR systems can use multi-channel speaker systems, such as 5.1 or 7.1 surround sound speakers, for audio output. The above SH representations can be used to derive per-channel reverberation times in any arbitrary multi-channel speaker system, given the positions of the individual speakers with respect to the user. The SH representations of the adjusted mean free path can then be evaluated at any speaker position (as per Equation 6.18)

to determine the reverberation time for the corresponding channel. Alternately, we can use the Ambisonics expressions for amplitude panning weights [60] to directly determine the contribution of the $l_{p,q}$ terms at each speaker position. For example, with first-order SH and N speakers, we use:

$$l_i = \frac{1}{N} \sum_j (1 - 2\omega_j \cdot \omega_i), \tag{6.20}$$

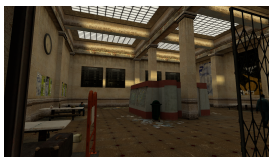
where $i \in [0, N-1]$ are the indices of the speakers, the indices j range over the number of rays traced from the listener, ω_j are the ray directions, and ω_i are the directions of the speakers relative to the listener. We can then evaluate a reverberation time for each speaker:

$$\mu_i = \beta l_i + (1 - \beta)\mu_0. \tag{6.21}$$

This enables realistic directional reverberation on a variety of speaker configurations, ranging from stereo to 5.1 or 7.1 home theater systems.

6.6 Results

We have integrated our approach into Valve's Source game engine. Sound is rendered using Microsoft's XAudio2 API. Ray tracing, mean free path estimation, proxy generation, and impulse response computation are performed continuously in a separate thread; the latest estimates are used to configure XAudio2's artificial reverberators for each channel as well as a per-channel convolution unit. Intel Math Kernel Library is used for convolution. All experiments were performed on an Intel Xeon X5560 with 4 cores and 2GB of RAM running Windows Vista; our implementation uses only a single CPU core. Figure 6.3 shows the benchmark scenes used in our experiments.



(a) Train Station (9k polygons)

(b) Citadel (23k polygons)

(c) Reservoir (32k polygons)

(d) Outlands (56k polygons)

Figure 6.3: Benchmark scenes used in our experiments.

Scene	Polygons	Ray Samples	Time (ms)
Train Station	9110	1024	7.88
Citadel	23231	2048	8.94
Reservoir	31690	1024	10.79
Outlands	55866	1024	4.59

Table 6.1: Performance of local distance average estimation.

These are indoor and outdoor scenes with dynamic objects (e.g. moving doors), as shown in the accompanying video.

6.6.1 Performance

Table 6.1 shows the time taken to perform the integration required to estimate mean free path. Our implementation uses the ray tracer built into the game engine, which is designed to handle only a few ray shooting queries arising from firing bullet weapons and from GUI picking operations; it is not optimized for tracing large batches of rays. Nonetheless, we observe high performance, indicating that our method is suitable for use in modern game engines running on current commodity hardware. Given the local distance average, the final mean free path and RT_{60} estimate is computed within 1–2 μs.

The complexity of the integration step is $O(k \log n)$, where k is the number of integration samples (rays) and n is the number of polygons in the scene. For low values of k, we expect very high performance with a modern ray tracer.

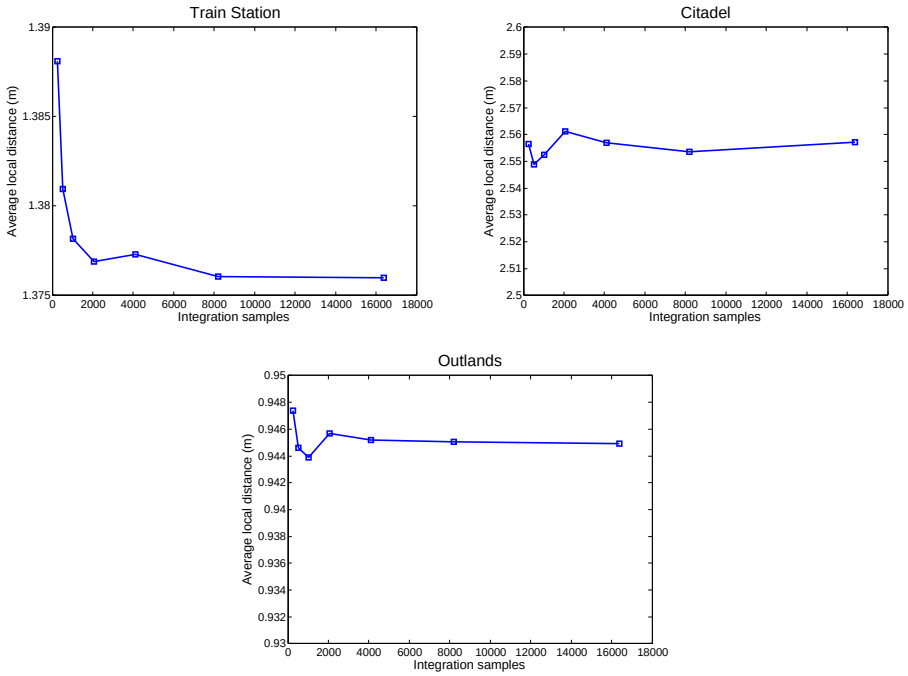


Figure 6.4: Convergence of local distance average estimate.

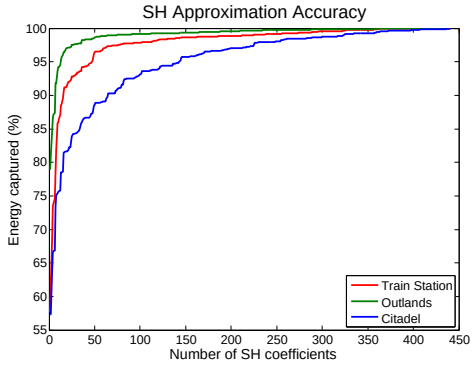


Figure 6.5: Accuracy of representing the local distance function in spherical harmonics, as a function of the number of SH coefficients.

6.6.2 Analysis

Figure 6.4 plots the estimated local distance average as a function of the number of rays traced from the listener, for different scenes. For clarity, the local distance average is computed by integrating over the unit sphere, without directional weights.

The plots demonstrate that tracing a large number of rays is not necessary; the local distance average quickly converges with only a small number of rays (1–2K); and can be evaluated very efficiently, even in large, complex scenes.

Figure 6.5 illustrates the accuracy of a spherical harmonics representation of the local distance function, for different scenes. The figure shows the percentage of energy captured in the spherical harmonics representation as a function of the number of coefficients, up to order 20 (i.e., $p = 20$). The figure clearly shows that very few coefficients are required to capture most of the directional variation ($75 - 80\%$).

Chapter 7

Aural Proxies

In this chapter, we describe an approach for efficiently simulating 2–4 orders of early reflections. The approach combines the image source method with simplified representations of the geometry around the listener. These simplified representations, called Aural Proxies [9], allow higher-order image sources to be constructed without performing multi-bounce ray tracing. The method complements the approach described in Chapter 6, allowing early reflections to also be computed within a few milliseconds, without requiring any precomputed data.

7.1 Image Source Method

State-of-the-art techniques for interactively modeling reflected sound are based on the image source method [4]. This method involves determining virtual *image sources* which represent reflected sound paths reaching the listener from the source. To determine the positions of the image sources, and which image sources contribute reflected sound to the listener, rays are traced from the source position, and recursively from each of the image sources.

7.1.1 Rectangular Rooms

Such multi-bounce ray tracing is possible in real-time [78] for up to around 4-5 orders of reflections. However, with all existing real-time ray tracers, achieving such a level of performance requires dedicating significant computational resources (a large number of CPU cores, or most, if not all, of the compute units on a GPU) solely to the audio pipeline. These computational demands cannot be practically met by modern game engines, that require most of the computational resources to be dedicated to rendering, physics simulation, or AI. Hence, we propose an approximate approach which demands significantly fewer computational resources.

Our approach only traces single-bounce rays, which can be used to compute image sources for first-order reflections. We next describe a local model for extrapolating from first-order image sources to higher-order image sources. This approach does not require tracing additional rays to compute higher-order reflections, and hence has a lower computational overhead than ray-tracing-based image source methods.

Our local model is based on the observation that in a rectangular (or *shoebox*) room, image sources are never occluded, and their positions can be computed by reflections about one of six planes, without having to trace any rays. In fact, in a rectangular room, the superposition of sound fields induced by the image sources obtained using this approach is an analytical solution of the wave equation in the scene [4].

7.2 Proxy Construction

We begin by fitting a shoebox to the local geometry around the listener (see Figure 7.1). We consider the hit points of all the ray traced from the listener during

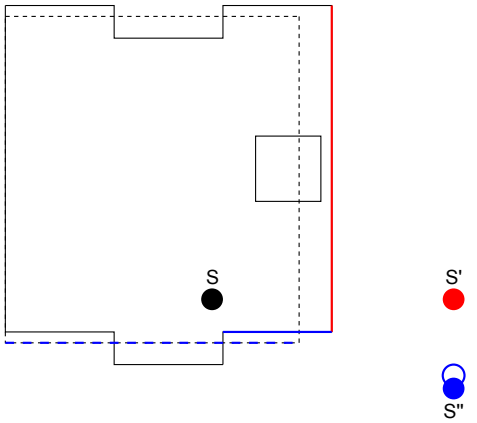


Figure 7.1: Higher-order reflections using a rectangular aural proxy. A source S is placed in a scene with walls and a rectangular object inside (solid lines). A ray-tracing-based image source method is used to construct the first-order image source S', by reflecting S about the surface shown in solid red. The aural proxy, shown with dashed lines, is used to reflect S' and construct the second-order image source S" (by reflecting S' about the plane of the blue dotted surface). No ray tracing is needed for the construction of S". The blue outline indicates the position of S" as computed by the ray-tracing-based image source method, by reflecting S' about the plane of the blue solid surface.

reverb estimation, and perform a cube map projection. This process bins each of the hit points to one of the six cube faces. Suppose the set of hit points binned to one particular cube face (with normal $\mathbf{n}$) is denoted by $\{d_i, \mathbf{n}_i, \alpha_i\}$, where d_i is the projection depth of the i^{th} hit point, $\mathbf{n}_i$ is the surface normal at the hit point, and α_i is the absorption coefficient of the surface at the hit point. We use this information to compute the following aggregate properties for the cube face:

Depth We average the depths of the hit points:

$$d = [d_i], \tag{7.1}$$

(where $[\cdot]$ denotes the averaging operator) to determine the average depth of the cube face from the listener along the appropriate coordinate axis.

Absorption We similarly average the absorption coefficients of the hit points:

$$\alpha = [\alpha_i], \tag{7.2}$$

to determine the absorption coefficient of the cube face. Note that this process automatically assigns higher weights to the absorption coefficients of surfaces with greater visible surface area (as seen from the listener's position).

Scattering In complex scenes, the surface normals $\mathbf{n}_i$ are likely to deviate to a varying extent from the cube face normal $\mathbf{n}$. Assuming the cube face to be perfectly planar is likely to result in excess reflected sound being computed. To address this issue, and allow the proxy geometry to better approximate the reflection and scattering behavior of the underlying scene geometry, we compute a scattering coefficient σ for the cube face, which describes the fraction of non-absorbed sound that is reflected in directions other than the specular reflection direction. Specifically, we compute the *random-incidence scattering coefficient*, which is defined as the fraction of reflected sound energy that is scattered away from the specular reflection direction, averaged over multiple incidence directions [89].

For any given incidence direction, a surface patch reflects sound in the specular direction for the cube face only if the local surface normal of the patch is aligned with the surface normal of the cube face. We define an alignment indicator function, $\chi_{\mathbf{n}}$, such that $\chi_{\mathbf{n}}(\mathbf{n}_i) = 1$ if and only if $||\mathbf{n} \cdot \mathbf{n}_i - 1|| \leq \epsilon$, and 0 otherwise, where ϵ is some suitably chosen tolerance. Since the total energy reflected from each hit point is $\sum_i (1 - \alpha_i)$, we get:

$$\sigma = 1 - \frac{\sum_i (1 - \alpha_i)\chi_{\mathbf{n}}(\mathbf{n}_i)}{\sum_i (1 - \alpha_i)}, \tag{7.3}$$

which we use as our scattering coefficient.

Note that we cannot use the listener's local coordinate axes for projection, since this would result in the shoebox dimensions varying even if the listener rotates in-place, resulting in an obvious instability in the reflected sound field. Hence, we use the world-space coordinate axes for projection.

7.3 Proxy-Based Reflections

Given the local shoebox proxy, we can quickly extrapolate from first-order reflections to higher-order reflections. We take the first-order image sources computed using ray tracing, and recursively reflect them about the faces of the proxy shoebox, yielding higher-order image sources. This process efficiently constructs approximate higher-order image sources. The image sources computed by this approach also have the important property that the directions of the higher-order image sources relative to the listener are plausibly approximated, i.e., if reflected sound is expected to be heard from the listener's right, the approximation tends to contain a reflection reaching the listener from the right. This is because geometry lying (say) to the right of the listener is mapped to a proxy face which also lies to the right of the listener. Therefore, the relative positions of two objects or surfaces roughly correspond to the relative positions of the proxy faces they are mapped to. (See the accompanying video for more.)

To account for absorption and surface normal variations, after each order of reflection, the strengths of the image sources are scaled by $(1 - \alpha)(1 - \sigma)$, where α is the absorption coefficient of the face about which the image source was reflected, and σ is its scattering coefficient.

7.4 Results

We have integrated our approach into Valve's Source game engine. Sound is rendered using Microsoft's XAudio2 API. Ray tracing, mean free path estimation, proxy generation, and impulse response computation are performed continuously in a separate thread; the latest estimates are used to configure XAudio2's artificial reverberators for each channel as well as a per-channel convolution unit. Intel Math Kernel Library is used for convolution. All experiments were performed on an Intel Xeon X5560 with 4 cores and 2GB of RAM running Windows Vista; our implementation uses only a single CPU core. Figure 6.3 shows the benchmark scenes used in our experiments. These are indoor and outdoor scenes with dynamic objects (e.g. moving doors), as shown in the accompanying video.

7.4.1 Performance

The time required to generate the proxy is scene-independent. In practice we observe around 0.9–1.0 ms for generating the proxy using 1024 samples; the cost scales linearly in the number of samples. Table 7.1 compares the performance of constructing higher-order image sources using our method to the time required by a reference ray-tracing-based image source method. The performance of our method is independent of scene complexity, whereas the image source method incurs increased computational overhead in complex scenes. Note that since both timings were measured by running the technique on complex models used for visual rendering, the reference times are particularly high. While these timings could be reduced by simplifying the model, the numbers highlight the fact that our approach can achieve high performance even on complex models designed for visual rendering, without necessitating an additional step in the designer's workflow where the model is simplified for acous-

Scene	Refl. Orders	Time (ms)	Ref. Time (ms)
Outlands	2	0.005	380
	3	0.010	3246
Reservoir	2	0.004	101
	3	0.009	656
Citadel	2	0.01	341
	3	0.02	3289
Train Station	2	0.005	30
	3	0.015	223
	4	0.049	1689

Table 7.1: Performance of proxy-based higher-order reflections, compared to reference image source method. Column 2 indicates the orders of reflection, Column 3 indicates time taken by our approach, and Column 4 indicates time taken by the ray-tracing-based image source method to compute the reference solution.

tic purposes.

7.4.2 Analysis

Figure 7.2 plots the estimated dimensions of the dynamically generated rectangular proxy as a function of the number of rays traced, for a given listener position in the Citadel scene. For example, the curve labeled "X" plots the difference (in meters) between the estimated world-space positions of the +X and -X faces of the proxy. The other two curves plot analogous quantities for the Y and Z axes. The plot shows that the estimated depths of the cube faces converge quickly, allowing us to trace fewer rays at run-time.

Figure 7.3 compares the impulse responses generated by our method against those generated by a reference ray-tracing-based image source method. In all cases, we computed up to 3 orders of reflection, with a maximum impulse response length of 2.0 seconds. For the reference image source method, we traced 16K primary rays from the source position, and 32 secondary rays recursively from each image source.

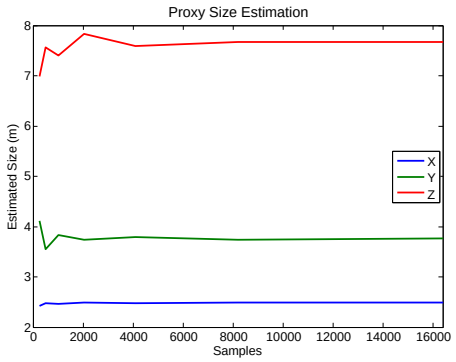


Figure 7.2: Convergence of proxy size estimation. The individual curves show the estimates for the X, Y, and Z dimensions of the proxy computed at a particular listener position in the Citadel scene.

For our method, we traced 16K primary rays from the source position to generate the rectangular proxy, which we then used to generate higher-order reflections. In all cases, the source and listener were placed at the same position.

In the case of the Train Station scene, our approach generates extraneous low-amplitude contributions, while retaining a similar overall decay profile. The larger number of contributions arises because our method maps many surfaces which do not actually contribute specular reflections at the listener to the same cube face. This leads to many more higher-order image sources being generated as compared to the reference method. The amplitudes of these contributions are lower since the estimated scattering coefficients compensate for the large variation in local surface normals over the proxy faces by reducing the amplitude of the reflected sound.

In the case of the Reservoir scene, our approach misses a reflection peak which can be seen in the reference impulse response (see Figure 7.3). This is most likely a higher-order reflection from one of the rocks (which are small relative to the rest of the scene). Our approach cannot model higher order reflections from relatively small, distinct features such as the rocks in this scene, since the dimensions of the rectangular proxy are dominated by the distant cliffs and terrain in this scene, which

occupy a larger visible projected surface area with respect to the listener position.

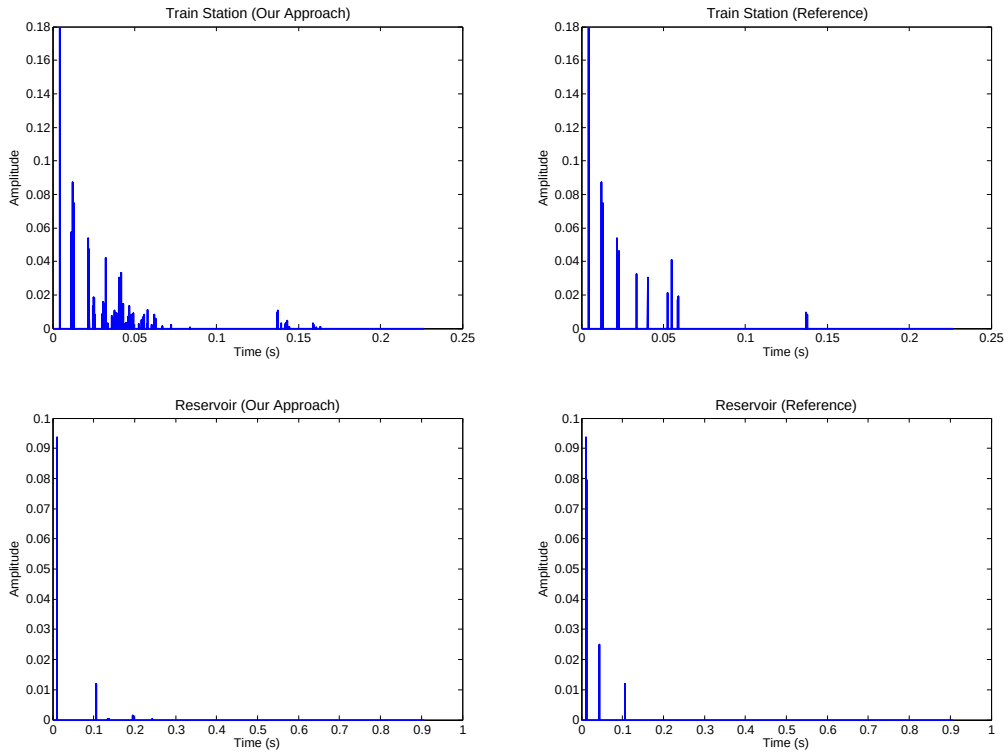


Figure 7.3: Comparison between impulse responses generated by aural proxies and a reference image source method.

7.4.3 Evaluation

We have performed a preliminary user study to compare the quality of early reflections generated by our approach against those generated by a reference ray-tracing-based image source method. The study involves 16 pairs of video clips showing the same sound clips (gunshots) rendered within an environment. For each of our benchmark scenes, we generated 4 pairs of sound clips. Two of these pairs contained one clip each from our method and the reference method. The remaining two pairs either contained two identical clips generated using the reference method, or two identical clips generated using our method. The ordering of clips was randomized for each

participant. For each pair of clips, participants were asked to rate a) which clip they considered more immersive, and b) which clip they thought matched better with the visual rendering. Both answers were given on a scale of 1 to 10, with 1 meaning the first clip in the pair was preferred strongly, and 10 meaning the second clip in the pair was preferred strongly.

Table 7.2 tabulates the results of this user study, gathered from 20 participants. Question 1 refers to the question regarding overall level of realism. Question 2 refers to the question regarding correlation with the visual rendering. For question and for each scene, the table provides the mean and standard deviation of the scores for three groups of questions. The first group, denoted REF/REF, contains video pairs containing two identical clips generated using the reference method. The second group, denoted OUR/OUR, contains video pairs containing two identical clips generating using our method. The third group, denoted REF/OUR, contains video pairs containing one clip generated using the reference method, and one clip generated using our method. In this group, low scores indicate a preference for the reference method, and high scores indicate a preference for our method.

As the results demonstrate, most participants did not exhibit a strong preference for either of the clips in any pair, since most of the mean scores are between 5 and 6. This indicates that the participants felt that our method generates results that are comparable to the reference method with respect to the subjective criteria of realism and correlation with visuals. On the other hand, the standard deviations may indicate that further research is warrented into the factors that affect a user's perception of sound propagation effects and their resulting level of immersion. However, this is a preliminary user study; we plan to perform a more extensive and detailed evaluation of our technique in the future.

Question	Scene	REF/REF		OUR/OUR		REF/OUR	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
1	Citadel	5.3	0.99	5.9	0.97	5.3	1.88
	Outlands	5.6	0.99	6.1	1.14	5.1	1.43
	Reservoir	5.8	1.29	6.0	2.11	5.5	2.35
	Train Station	6.2	1.36	6.2	1.09	5.6	2.13
2	Citadel	5.3	1.24	5.8	1.06	5.5	2.02
	Outlands	5.6	0.83	6.0	1.02	5.4	1.43
	Reservoir	5.8	1.33	5.7	2.13	5.2	2.26
	Train Station	6.1	1.43	5.8	1.21	5.3	1.98

Table 7.2: Results of our preliminary user study. For each question and for each scene, we tabulate the mean and standard deviations of the responses given by the participants. The columns labelled REF/REF are the scores for questions involving comparisons between two identical clips generated using the reference image source method. The columns labelled OUR/OUR are the scores for questions involving comparisons between two identical clips generated using our approach. The columns labelled REF/OUR are the scores for questions involving comparisons between our approach and the reference approach.

Chapter 8

Conclusion

In this thesis, we have presented multiple algorithms for simulating sound propagation in interactive applications. We have presented precomputation-based algorithms that use frequency-domain or time-domain acoustic transfer operators to obtain realistic, geometry-dependent results. We have also presented efficient algorithms based on simplified propagation models for reverberation and early reflections.

In this chapter, we summarize our results, discuss limitations, and present avenues for future work.

8.1 Frequency-Domain Diffuse Acoustic Transfer

We have described a precomputed direct-to-indirect transfer approach to solving the acoustic rendering equation in the frequency domain for diffuse reflections. We have demonstrated that our approach is able to efficiently simulate diffuse reflections for a moving sources and listeners in static scenes. In comparison with existing methods, our approach offers a significant performance advantage when handling moving sources.

Limitations Since this approach is a precomputation-based algorithm, it cannot be used for scenes with dynamic objects. In such situations, ray-tracing-based algorithms are the best available choice. However, in many applications, including games and virtual environments, scenes are mostly static, with relatively few moving parts, hence our algorithm can be used to model reflections within the static portions of the scene.

Our algorithm performs matrix-vector multiplications on large matrices at runtime. The matrix size depends on the surface area of the scene and the number of geometric primitives used to represent the scene. Therefore, our method is useful mainly for scenes of low to medium complexity.

Another limitation arises from the approach we use [68] to reconstruct the energy response from the subsampled Fourier coefficients. The replication of Fourier coefficients leads to comb-filter artifacts in the final audio, and is an inherent limitation of the reconstruction approach. An alternative would be to treat the Fourier coefficients as defining the envelope of a noise process [72]. Both these approaches are prone to errors; further study is needed to determine the suitability of one over the other based on empirical and perceptual error.

Finally, the transfer matrix is computed using the acoustic rendering equation [67], which has its own limitations, in that it is an energy-based approach (and hence cannot easily model interference) and is based on geometric approximations to the acoustic wave equation (and hence cannot accurately model low-frequency wave effects such as diffraction).

Future Work In most complex scenes, each surface sample may influence only a few other samples, due to occlusions. We could subdivide the scene into cells separated by portals, compute transfer operators for each cell independently, and

model the interchange of sound energy at the portal boundaries. Cells and portals have been previously used to model late reverberation [72], and would be a promising research direction for acoustic radiance transfer.

The acoustic response is typically a smooth function over the surfaces of the scene. Therefore, it would be beneficial to exploit spatial coherence by projecting the transfer operator into basis functions (such as wavelets) defined over the surfaces of the scene.

Some radiance transfer algorithms in visual rendering [71, 31] can model glossy reflections by using a directional basis such as spherical harmonics (SH) at each surface sample. Such a strategy can also be applied to model glossy reflections and diffraction of sound, however, the memory requirements for such an approach might be prohibitive.

Fractional delays [41] may also be used to generate a more accurate impulse response when propagation path delays do not lie exactly on time samples.

8.2 Compact Acoustic Transfer Operators

We have presented an efficient algorithm for computing sound propagation for interactive applications. The algorithm is a two-pass hybrid of ray tracing and radiance transfer algorithms. We have demonstrated that the algorithm can model high orders of reflection (specular as well as diffuse) and edge diffraction at near-interactive rates with low memory overhead.

Limitations Our approach is based on geometric sound propagation using path tracing. As a result, all the limitations of geometric acoustics apply to our method. In particular, our approach cannot accurately model low-frequency reflections and edge

diffraction. Since the acoustic rendering equation is an energy-based formulation of sound propagation [67], phase-related effects, such as some cases of interference, cannot be modeled.

The run-time ray-tracing pass only computes early reflections (2–3 orders). Coupled with the fact that the radiance transfer pass assumes all surface samples are diffuse emitters, this implies that we cannot model higher-order purely specular reflections (such as flutter echoes) or higher-order purely diffracted paths (such as diffraction around complex curved surfaces).

Our handling of dynamic objects is restricted to modeling direct "shadows" cast by a moving source due to moving objects. Since the transfer operators are computed over static surfaces only, we cannot model "indirect shadows", i.e., occlusion of reflected sound by moving objects. As a result, we cannot completely handle situations such as moving doors between static rooms.

Like other precomputation algorithms [83, 63], our approach performs significant compression of the precomputed data in order to run on commodity hardware. This compression is lossy, and results in a reduction in the accuracy of the simulation results. As a result, our algorithm is not practical for detailed room acoustical analysis. It is designed for games and other interactive applications where approximate directional cues, echoes and reverberation can be dynamically updated to generate a plausible, immersive audio experience.

Future Work Our algorithm can serve as the basis for much future research geared towards providing immersive audio in games and interactive applications. Firstly, it would be useful to investigate the possibility of using spherical harmonics to model the directional variation of acoustic radiance, while keeping memory overheads low. Another strategy for reducing memory requirements might be based on the structure

of the acoustic transfer operator: in most game environments, occlusion would lead to clustering within the transfer matrix. These clusters would roughly correspond to cells in a cells-and-portals subdivision of the scene. Therefore, it might be useful to consider computing a per-cell acoustic transfer operator and modeling sound propagation between cells via the portals. The clusters may also be useful in optimizing the distribution of surface samples. In general, developing techniques for automatically choosing surface sample pairs for computing example echograms would be an interesting avenue for future research.

Since nearest-neighbor mapping is used for the hit-points of gather rays, there may be temporal error in interpolating echograms. This may lead to errors in the final echogram. It would be useful to develop delay-aware interpolation schemes to address these issues.

It would be very interesting to extend our basic precomputation framework to a pressure-based formulation by computing the sample-to-sample transfer operators using a numerical solver for the acoustic wave equation. This would essentially amount to a precomputation-based Monte Carlo solution of the Kirchoff integral formulation of the wave equation, and would result in increased accuracy in modeling wave phenomena such as diffraction. A related formulation of transfer operators have been used for numerically solving the Helmholtz equation using the Equivalent Source Method [52].

Finally, it would be very useful to integrate our approach with a precomputation-based sound synthesis algorithm such as Precomputed Acoustic Transfer (PAT) [34] to develop a unified approach for performing efficient propagation of synthesized sound in interactive applications.

8.3 Aural Proxies and Ambient Reverberance

We have presented an efficient technique for approximately modeling sound propagation effects in indoor and outdoor scenes for interactive applications. The technique is based on adjusting user-controlled reverberation parameters in response to the listener's movement within a virtual world, as well as a local shoebox proxy for generating early reflections with a plausible directional distribution. The technique generates immersive directional reverberation and reflection effects, and can easily scale to multi-channel speaker configurations. It is easy to implement and can be easily integrated into any modern game engine, without significantly re-architecting the audio pipeline, as demonstrated by our integration with Valve's Source engine.

Limitations Our reverberation approach does not account for spatially-varying surface absorption properties; however, this is a limitation of the underlying statistical model, the Eyring model [23]. Our approach for modeling reflections involves a coarse shoebox proxy; as a result the accuracy of the generated higher-order reflections depends on how good a match the proxy model is to the underlying scene geometry. Finally, since our reverberation approach does not perform global (multi-bounce) ray tracing, but involves an user-controlled reverberation time, it is subject to error in the adjusted mean free path.

Future Work There are many avenues for future work. One main challenge is to develop a method for incorporating multi-bounce ray tracing into mean free path estimation in real-time, so as to generate more realistic reverberation. Our current approach for reverberation estimation does not account for diffracted rays reaching the listener; incorporating such rays would result in a richer frequency-dependent

variation in the reverberation. The reverberation approach also does not account for the scattering properties of the surfaces hit by rays. One approach for modeling scattering properties would be to trace secondary rays from the hit points, and estimate distances to scene surfaces from the hit point. This way, if the sound at a hit point tends to be scattered into a larger room, we would obtain more reverb from the direction of the hit point. This approach would require more expensive Monte Carlo tracing (and would be even closer to ambient occlusion than the present technique), hence for performance reasons, approximate techniques analogous to screen-space ambient occlusion [66] may need to be developed. Furthermore, it would be interesting to explore a more accurate approach for fitting shoebox proxies to scene geometry, based on projections along the principal axes of the point cloud of geometry samples obtained through ray tracing. Finally, we need to evaluate our approach in more game and VR scenarios and perform detailed user studies to evaluate its benefits.

8.4 Trade-offs

The algorithms based on Precomputed Acoustic Radiance Transfer (Chapters 3–5) account for the geometry of the entire scene. This makes them suitable for interactive applications were increased accuracy is desirable in exchange for somewhat reduced performance. We have also demonstrated that PART can be used to simulate approximate diffraction effects (Chapter 5). On the other hand, the Ambient Reverberance and Aural Proxies algorithm (Chapters 6 and 7) only account for the geometry around the listener. This makes them suitable for applications where performance is of utmost importance, and where plausible variation in sound propagation effects is sufficient. These methods can handle dynamic scenes, but do not support diffraction or complex multiple scattering effects.

8.5 Future Work

One of the most important topics for future research into precomputation-based sound propagation algorithms is the development of more compact representations for radiance (or other similar precomputed data). On the other hand, the overall goal of further research into simplified statistical models for sound propagation is that of improved accuracy. We believe that the combination of these two ideas – source- and listener-independent precomputation, and simplified propagation models – offers the best of both worlds, and presents one of the most promising avenues of future work in the area of sound propagation for interactive applications.

Other topics for future research include techniques for automatically simplifying geometric models to retain only the acoustically relevant features, as well as hybrid techniques for combining numerical and geometric propagation techniques, thus providing a balance between simulation accuracy and performance in large, complex scenes.

Bibliography

[1] 2K Games. Bioshock, 2007.

[2] Ahmed, N., Natarajan, T., and Rao, K. R. Discrete cosine transform. *IEEE Trans. Computers C-23*, 1 (1974), 90–93.

[3] Allen, J. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on 25*, 3 (1977), 235–238.

[4] Allen, J. B., and Berkley, D. A. Image method for efficiently simulating small-room acoustics. *J. Acoustical Society of America 65*, 4 (1979), 943–950.

[5] An, S., James, D. L., and Marschner, S. Motion-driven concatenative synthesis of cloth sounds. *ACM Trans. Graphics 31*, 4 (2012).

[6] Antani, L., Chandak, A., Savioja, L., and Manocha, D. Interactive sound propagation using compact acoustic transfer operators. *ACM Trans. Graphics 31*, 1 (2012).

[7] Antani, L., Chandak, A., Taylor, M., and Manocha, D. Efficient finite-edge diffraction using conservative from-region visibility. *Applied Acoustics 73*, 3, 218–233.

[8] Antani, L., Chandak, A., Taylor, M., and Manocha, D. Direct-to-indirect acoustic radiance transfer. *IEEE Trans. Visualization and Computer Graphics 18*, 2 (2012), 261–269.

[9] Antani, L., and Manocha, D. Aural proxies and directionally-varying reverberation for interactive sound propagation in virtual environments. *IEEE Trans. Visualization and Computer Graphics 19*, 4 (2013), 567–575.

[10] Begault, D. *3D Sound for Virtual Reality and Multimedia*. Academic Press, 1994.

[11] Bertram, M., Deines, E., Mohring, J., Jegorovs, J., and Hagen, H. Phonon tracing for auralization and visualization of sound. In *IEEE Visualization 2005* (2005), pp. 151–158.

[12] Borish, J. Extension of the image model to arbitrary polyhedra. *J. Acoustical Society of America 75*, 6 (1984), 1827–1836.

[13] Botteldooren, D. Finite-difference time-domain simulation of low-frequency room acoustic problems. *J. Acoustical Society of America 98*, 6 (1995), 3302–3308.

[14] Bracewell, R. N. *The Fourier transform and its applications*. McGraw-Hill, 2000.

[15] Brigham, E. O. *The Fast Fourier Transform and its applications*. Prentice-Hall, 1988.

[16] Chadwick, J., An, S., and James, D. L. Harmonic shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graphics 28*, 5 (2009), 119:1–119:10.

[17] Chandak, A., Antani, L., Taylor, M., and Manocha, D. Fastv: From-point visibility culling on complex models. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 28* (2009), 1237–1246.

[18] Chandak, A., Lauterbach, C., Taylor, M., Ren, Z., and Manocha, D. Ad-frustum: Adaptive frustum tracing for interactive sound propagation. In *Proc. IEEE Visualization* (2008).

[19] Christensen, C. L. *ODEON Room Acoustics Program User Manual*. Odeon A/S, 2009.

[20] Ciskowski, R. D., and Brebbia, C. A. *Boundary element methods in acoustics*. Springer, 1991.

[21] Eidos Interactive. Thief: Deadly shadows, 2004.

[22] Engquist, B., and Runborg, O. Computational high frequency wave propagation. *Acta Numerica 12* (2003), 181–266.

[23] Eyring, C. F. Reverberation time in dead rooms. *J. Acoustical Society of America 1* (1930), 217–241.

[24] Farina, A. Ramsete: A new pyramid tracer for medium and large scale acoustic problems. In *Proc. EURONOISE* (1995).

[25] Foale, C., and Vamplew, P. Portal-based sound propagation for first-person computer games. In *Australasian Conference on Interactive Entertainment 2007* (2007), pp. 9:1–9:8.

[26] Frictional Games. Amnesia: The dark descent, 2010.

[27] Funkhouser, T., Carlbom, I., Elko, G., Pingali, G., Sondhi, M., and West, J. A beam tracing approach to acoustic modeling for interactive virtual environments. In *SIGGRAPH 1998* (1998), pp. 21–32.

[28] Gardner, W. G., and Martin, K. D. Hrtf measurements of a kemar. *J. Acoustical Society of America 97*, 6 (1995).

[29] Gerardi, M., Rothbaum, B. O., Ressler, K., Heekin, M., and Rizzo, A. Virtual reality exposure therapy using a virtual iraq: case report. *Journal of Traumatic Stress 21*, 2 (2008), 209–213.

[30] Gumerov, N. A., and Duraiswami, R. A broadband fast multipole accelerated boundary element method for the three dimensional helmholtz equation. *J. Acoustical Society of America 125*, 1 (2009), 191–205.

[31] Hašan, M., Pellacini, F., and Bala, K. Direct-to-indirect transfer for cinematic relighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) 25*, 3 (2006), 1089–1097.

[32] Heckbert, P. S. Adaptive radiosity textures for bidirectional ray tracing. In *Proc. SIGGRAPH* (1990).

[33] IASIG. Interactive 3d audio rendering guidelines, level 2.0. `http://www.iasig.org/pubs/3dl2v1a.pdf`, 1999.

[34] James, D. L., Barbi, J., and Pai, D. K. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. In *SIGGRAPH* (2006).

[35] Jot, J.-M., and Chaigne, A. Digital delay networks for designing artificial reverberators. In *AES Convention* (1991).

[36] Kouyoumjian, R., and Pathak, P. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proceedings of the IEEE 62*, 11 (1974), 1448–1461.

[37] Krokstad, A., Strom, S., and Sorsdal, S. Calculating the acoustical room response by the use of a ray tracing technique. *J. Sound and Vibration 8*, 1 (1968), 118–125.

[38] Kuttruff, H. *Room Acoustics*. Elsevier Science Publishing Ltd., 1991.

[39] Kuttruff, H. A simple iteration scheme for the computation of decay constants in enclosures with diffusely reflecting boundaries. *The Journal of the Acoustical Society of America 98*, 1 (1995), 288–293.

[40] Kuttruff, H. K. Auralization of Impulse Responses Modeled on the Basis of Ray-Tracing Results. *Journal of the Audio Engineering Society 41*, 11 (November 1993), 876–880.

[41] Laakso, T. I., Välimäki, V., Karjalainen, M., and Laine, U. K. Splitting the unit delay - tools for fractional delay filter design. *IEEE Signal Processing Magazine 13* (1996).

[42] Laine, S., Siltanen, S., Lokki, T., and Savioja, L. Accelerated beam tracing algorithm. *Applied Acoustics 70*, 1 (2009), 172–181.

[43] Lauterbach, C., Chandak, A., and Manocha, D. Interactive sound rendering in complex and dynamic scenes using frustum tracing. In *Proc. IEEE Visualization* (2007).

[44] Lauterbach, C., Yoon, S.-E., Tuft, D., and Manocha, D. Rt-deform: Interactive ray tracing of dynamic scenes using bvhs. In *IEEE. Symp. Interactive Ray Tracing* (2006).

[45] Lehtinen, J., Zwicker, M., Turquin, E., Kontkanen, J., Durand, F., Sillion, F. X., and Aila, T. A meshless hierarchical representation for light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008) 27*, 3 (2008), 1–9.

[46] Lentz, T., Schroeder, D., Vorlander, M., and Assenmacher, I. Virtual reality system with integrated sound field simulation and reproduction. *EURASIP Journal of Applied Signal Processing 2007*, 1 (2007).

[47] Lévy, B., Petitjean, S., Ray, N., and Maillot, J. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (2002), 362–371.

[48] Liu, Q. H. The pseudospectral time-domain (pstd) method: A new algorithm for solutions of maxwell's equations. In *Proc. IEEE Antennas and Propagation Society International Symp.* (1997).

[49] Loève, M. *Probability Theory Vol. II.* Springer-Verlag, 1978.

[50] Malham, D. G. Ambisonics: A technique for low-cost, high-precision three-dimensional sound diffusion. In *Proc. International Computer Music Conf.* (1990).

[51] McGookin, D., Brewster, S., and Priego, P. Audio bubbles: Employing non-speech audio to support tourist wayfinding. *Haptic and Audio Interaction Design (LNCS 5763)* (2009), 41–50.

[52] Mehra, R., Raghuvanshi, N., Antani, L., Chandak, A., Curtis, S., and Manocha, D. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Trans. Graphics* (to appear).

[53] Melzer, A., Kindsmuller, M., and Herczeg, M. Audioworld: A spatial audio tool for acoustic and cognitive learning. *Haptic and Audio Interaction Design (LNCS 6306)* (2010), 46–54.

[54] Moss, W., Yeh, H., Hong, J.-M., Lin, M. C., and Manocha, D. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Trans. Graphics 29*, 3 (2010).

[55] Ochmann, M. The source simulation technique for acoustic radiation problems. *Acustica 81* (1995), 512–527.

[56] Okada, M., Onoye, T., and Kobayashi, W. A ray tracing simulation of sound diffraction based on the analytic secondary source model. *IEEE Trans. Audio, Speech, and Language Processing 20*, 9, 2448–2460.

[57] Parker, S., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., and Stich, M. Optix: A general purpose ray tracing engine. In *SIGGRAPH* (2010).

[58] Pielot, M., Henze, N., Heuten, W., and Boll, S. Tangible user interface for the exploration of auditory city maps. *Haptic and Audio Interaction Design (LNCS 4813)* (2007), 86–97.

[59] Pierce, A. D. *Acoustics: An introduction to its physical principles and applications*. Acoustical Society of America, 1989.

[60] Pulkki, V. *Spatial sound generation and perception by amplitude panning techniques*. PhD thesis, Helsinki University of Technology, 2001.

[61] Raghuvanshi, N., and Lin, M. C. Interactive sound synthesis for large scale environments. In *Symposium on Interactive 3D Graphics and Games (I3D)* (2006).

[62] Raghuvanshi, N., Narain, R., and Lin, M. C. Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Trans. Visualization and Computer Graphics 15*, 5 (2009), 789–801.

[63] Raghuvanshi, N., Snyder, J., Mehra, R., Lin, M., and Govindaraju, N. Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4 (2010), 68:1 – 68:11.

[64] Ren, Z., Yeh, H., and Lin, M. C. Synthesizing contact sounds between textured objects. In *IEEE Virtual Reality* (2010).

[65] Reshetov, A., Soupikov, A., and Hurley, J. Multi-level ray tracing algorithm. In *SIGGRAPH* (2005).

[66] Shanmugam, P., and Arikan, O. Hardware accelerated ambient occlusion techniques on gpus. In *Proc. Symposium on Interactive 3D Graphics* (2007).

[67] Siltanen, S., Lokki, T., Kiminki, S., and Savioja, L. The room acoustic rendering equation. *Journal of the Acoustical Society of America 122*, 3 (2007), 1624–1635.

[68] Siltanen, S., Lokki, T., and Savioja, L. Frequency domain acoustic radiance transfer for real-time auralization. *Acta Acustica united with Acustica 95* (2009), 106–117.

[69] Siltanen, S., Lokki, T., Savioja, L., and Christensen, C. L. Geometry reduction in room acoustics modeling. *Acta Acustica united with Acustica 94* (2008), 410–418.

[70] Sloan, P.-P. Stupid spherical harmonics tricks. In *Game Developers Conference* (2008).

[71] Sloan, P.-P., Kautz, J., and Snyder, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002) 21*, 3 (2002), 527–536.

[72] Stavrakis, E., Tsingos, N., and Calamia, P. Topological sound propagation with reverberation graphs. *Acta Acustica united with Acustica* (2008).

[73] Stephenson, U. M. An analytically derived sound particle diffraction model. *Acta Acustica united with Acustica 96* (2010), 1051–1068.

[74] Stephenson, U. M., and Svensson, U. P. An improved energetic approach to diffraction based on the uncertainty principle. *19th International Congress on Acoustics (ICA)* (2007).

[75] Summers, J. E., Torres, R. R., and Shimizu, Y. Statistical-acoustics models of energy decay in systems of coupled rooms and their relation to geometrical acoustics. *Journal of the Acoustical Society of America 116*, 2 (2004), 958–969.

[76] Svensson, U. P., Fred, R. I., and Vanderkooy, J. An analytic secondary source model of edge diffraction impulse responses. *Journal of the Acoustical Society of America 106* (1999), 2331–2344.

[77] Taflove, A., and Hagness, S. C. *Computational Electrodynamics: The finite-difference time-domain method*, third ed. Artech House, 2005.

[78] Taylor, M., Chandak, A., Mo, Q., Lauterbach, C., Schissler, C., and Manocha, D. Guided multiview ray tracing for fast auralization. *IEEE Trans. Visualization and Computer Graphics 18*, 11 (2012), 1797–1810.

[79] Taylor, M., Chandak, A., Ren, Z., Lauterbach, C., and Manocha, D. Fast edge diffraction for sound propagation in complex virtual environments. In *Proc. EAA Symp. Auralization* (2009).

[80] Taylor, M. T., Chandak, A., Antani, L., and Manocha, D. Resound: interactive sound rendering for dynamic virtual environments. In *ACM Multimedia 2009* (2009), pp. 271–280.

[81] Thompson, L. L. A review of finite-element methods for time-harmonic acoustics. *J. Acoustical Society of America 119*, 3 (2006), 1315–1330.

[82] Tsingos, N. Perceptually-based auralization. In *International Congress on Acoustics* (2007).

[83] Tsingos, N. Precomputing geometry-based reverberation effects for games. In *Audio Engineering Society Conference: Audio for Games* (2009).

[84] Tsingos, N., Dachsbacher, C., Lefebvre, S., and Dellepiane, M. Instant sound scattering. In *Proc. Eurographics Symposium on Rendering* (2007).

[85] Tsingos, N., Funkhouser, T., Ngan, A., and Carlbom, I. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH 2001* (2001), pp. 545–552.

[86] Veach, E. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1997.

[87] Vecherin, S. N., Wilson, D. K., and Ostashev, V. E. Incorporating source directionality into outdoor sound propagation calculations. *J. Acoustical Society of America 130*, 6 (2011), 3608–3622.

[88] Vorlander, M. Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm. *Journal of the Acoustical Society of America 86*, 1 (1989), 172–178.

[89] Vorlander, M., and Mommertz, E. Definition and measurement of random-incidence scattering coefficients. *Applied Acoustics 60*, 2 (2000), 187–199.

[90] Wald, I., Slusallek, P., Benthin, C., and Wagner, M. Interactive rendering with coherent ray tracing. In *Eurographics* (2001).

[91] Wallace, J. R., Cohen, M. F., and Greenberg, D. P. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (Proceedings of SIGGRAPH 1987) 21*, 4 (1987), 311–320.

[92] Wang, Ye; Vilermo, M. Modified discrete cosine transform: Its implications for audio coding and error concealment. *J. Audio Eng. Soc 51*, 1/2 (2003), 52–61.

[93] White, G. R., Fitzpatrick, G., and McAllister, G. Toward accessible 3d virtual environments for the blind and visually impaired. In *Proc. International Conference on Digital Interactive Media in Entertainment and Arta (DIMEA)* (2008).

[94] Zheng, C., and James, D. L. Harmonic fluids. In *SIGGRAPH* (2010).

[95] Zheng, C., and James, D. L. Rigid-body fracture sound with precomputed soundbanks. In *SIGGRAPH* (2010).

[96] Zhukov, S., Inoes, A., and Kronin, G. An ambient light illumination model. *Rendering Techniques* (1998).