EFFICIENT INTERACTIVE SOUND PROPAGATION IN DYNAMIC ENVIRONMENTS

Carl Schissler

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 in the College of Arts and Sciences.

Chapel Hill
2017

Approved by:

Dinesh Manocha

Ming C. Lin

Ravish Mehra

Gary Bishop

Turner Whitted

©2017
Carl Schissler
ALL RIGHTS RESERVED

ABSTRACT

CARL SCHISSLER: Efficient Interactive Sound Propagation in Dynamic Environments
(Under the direction of Dinesh Manocha)

The physical phenomenon of sound is ubiquitous in our everyday life and is an important component of immersion in interactive virtual reality applications. Sound propagation involves modeling how sound is emitted from a source, interacts with the environment, and is received by a listener. Previous techniques for computing interactive sound propagation in dynamic scenes are based on geometric algorithms such as ray tracing. However, the performance and quality of these algorithms is strongly dependent on the number of rays traced. In addition, it is difficult to acquire acoustic material properties. It is also challenging to efficiently compute spatial sound effects from the output of ray tracing-based sound propagation. These problems lead to increased latency and less plausible sound in dynamic interactive environments.

In this dissertation, we propose three approaches with the goal of addressing these challenges. First, we present an approach that utilizes temporal coherence in the sound field to reuse computation from previous simulation time steps. Secondly, we present a framework for the automatic acquisition of acoustic material properties using visual and audio measurements of real-world environments. Finally, we propose efficient techniques for computing directional spatial sound for sound propagation with low latency using head-related transfer functions (HRTF).

We have evaluated both the performance and subjective impact of these techniques on a variety of complex dynamic indoor and outdoor environments and observe an order-of-magnitude speedup over previous approaches. The accuracy of our approaches has been validated against real-world measurements and previous methods. The proposed techniques enable interactive simulation of sound propagation in complex multi-source dynamic environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BRDF	Bidirectional Reflection Distribution Function
C_{80}	Clarity (ratio of early to late sound energy in decibels)
CNN	Convolutional Neural Network
CPU	Central Processing Unit
D_{50}	Definition (ratio of early to late sound energy, $\%$)
DPC	Diffuse Path Cache
EDT	Early Decay Time
ER	Early Reflections
FIR	Finite Impulse Response
FFT	Fast Fourier Transform
HRTF	Head-Related Transfer Function
IIR	Infinite Impulse Response
IR	Impulse Response
IRC	Impulse Response Cache
ISM	Image Source Method
LAB	CIE LAB perceptual color space
LR	Late Reverberation
MINC	Materials in Context
RGB	Red Green Blue (color space)
RGB-D	Red Green Blue with Depth (color space)
RISM	Ray-based Image Source Method
RT_{60}	Reverberation Time (the time for pressure to decay by 60 dB)
SH	Spherical Harmonics
SIFT	Scale Invariant Feature Transform
SNR	Signal to Noise Ratio
SPC	Specular Path Cache
SPL	Sound Pressure Level
TS	Center Time

LIST OF SYMBOLS

Math Functions:

$Y_{lm}(\vec{x})$	Spherical harmonic (SH) basis functions as a function of direction $\vec{x}$
n	Spherical harmonic order
$\delta(x)$	Dirac delta function

Counts, Objects:

N_i	Number of sound propagation paths
N_r	Number of primary rays emitted from source or listener
N_T	Number of triangles in the environment
N_S	Number of sound sources in the environment
N_L	Number of listeners in the environment
T_j	The jth triangle in the environment
S_k	The kth sound source in the environment
L_l	The lth listener in the environment
d	Acoustic interaction order
d_{max}	Maximum acoustic interaction order

Sound Attributes:

r	Sound propagation path length in meters
t	Time, in seconds
ω	Frequency
$\bar{\omega}$	Frequency band centered at ω
t_i	Delay time for sound propagation path i
$\vec{x}_i$	Incoming world-space sound propagation direction at the listener for path i
$X(\vec{x})$	Distribution of incoming sound intensity at the listener for frequency band $\bar{\omega}$
$X_{lm\bar{\omega}}$	Spherical harmonic coefficients for distribution of intensity for band $\bar{\omega}$
$I_{i\bar{\omega}}$	Sound intensity for propagation path i and frequency band $\bar{\omega}$
$I_{\bar{\omega}}(t)$	Sound intensity impulse response for frequency band $\bar{\omega}$
I_0	Reference sound intensity, 10^{-12}W
p_0	Reference sound pressure, 2×10^{-5}Pa

c	Speed of sound in propagation medium
z_0	Characteristic specific acoustic impedance of the propagation medium
P	Sound power, in watts
$T_q(\omega)$	Absolute threshold of human hearing as a function of frequency ω

Temporal Coherence:

K	Specular path identifier, sequence of triangle reflections
λ	Triangle subdivision patch size
(ζ, ξ)	Triangle subdivision patch location identifier
$\vec{q}$	Ray-triangle intersection point
$(\gamma_0, \gamma_1, \gamma_2)$	Baycentric coordinates of $\vec{q}$
η	Number of rays that reach the listener for a diffuse reflection path cache entry
μ	Number of rays emitted from the source while diffuse cache entry was valid
τ	Temporal coherence smoothing time constant, in seconds
α	Exponential smoothing factor determined from τ.

Material Optimization:

R	Sound intensity reflection coefficient
α	Sound pressure absorption coefficient for random sound incidence
b	Impulse response intensity histogram bin
m	Material patch
$W_{\bar{\omega}}$	Acoustic optimization weight matrix for frequency band $\bar{\omega}$
$w_{\bar{\omega}bm}$	Entry of weight matrix corresponding to IR bin b and material m
$p^S(t)$	Optimization simulated pressure impulse response
$p^T(t)$	Optimization target pressure impulse response
$I^S_{\bar{\omega}}(t)$	Intensity-time curve for optimization simulated IR
$I^T_{\bar{\omega}}(t)$	Intensity-time curve for optimization target IR
$I^S_{\bar{\omega}b}$	Intensity-time histogram for optimization simulated IR
$I^T_{\bar{\omega}b}$	Intensity-time histogram for optimization target IR
$E_{\bar{\omega}b}$	Error in decibels between $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$

Sound Rendering, Spatial Sound:

$p(t)$	Pressure impulse response as a function of time
$p_L(t)$	Pressure impulse response for the left ear
$p_R(t)$	Pressure impulse response for the right ear
$s(t)$	Anechoic sound source audio in units of pascals
$q_L(t)$	Sound source audio received by listener's left ear
$q_R(t)$	Sound source audio received by listener's right ear
$H(\vec{x}, t)$	Head-related transfer function as a function of direction $\vec{x}$ and time t
$h_{lm}(t)$	Head-related transfer function in the SH domain as a function of time t
$h_{lm}(\omega)$	Head-related transfer function in the SH domain as a function of frequency ω
L	Impulse response partition size

CHAPTER 1: INTRODUCTION

One of the most important human senses is our sense of hearing, the ability to perceive sound. Sound is a pressure wave produced by the vibration of an acoustic medium such as air or water. As humans, we can hear sound waves with frequencies between 20Hz and $20,000$Hz. Sound below this frequency range is called *infrasound*, while sound above this frequency range is called *ultrasound*. Hearing is the second most important sense that we use to interact with the environment (Blauert, 1997). Soundhas been a primary medium for the development of language, music, and possibly human intelligence. In the present day, sound remains an important part of everyday life as a vehicle for communication, education, and entertainment.

Sound propagation is the process by which pressure waves are emitted from a source into the propagation medium, transmitted and reflected through the environment, and eventually heard by a listener. When sound is produced, it can have various interactions with the environment. These include reflection, scattering, diffraction, air absorption, and refraction. During sound propagation, sound is delayed and attenuated in ways that provide a listener information about the environment and sound sources. The effects of sound propagation are responsible for familiar phenomena like echoes, reverberation, and Doppler shifting, as well as the subjective audible differences between environments like a gymnasium, office, forest, or cathedral.

1.1 Sound Propagation

Sound is a wave consisting of alternating regions of high and low pressure in a sound propagation medium. The distance between successive pressure highs or lows is the wavelength. The wavelength for sound at audible frequencies in air ranges from about 1.7cm to over 17m. Sound is emitted from a source when small surface vibrations cause the surrounding medium to vibrate in the form of pressure waves. These waves then propagate from the source at the speed of sound, $c \approx 343$m/s in air at room temperature. If the sound source is omnidirectional, the same pressure waves are radiated in all directions. However, most sound sources including the human voice are highly directional and radiate sound in a frequency and direction-dependent manner.

After sound is emitted from the source, it propagates through the environment. When sound interacts with an object in the environment, it may be reflected, scattered, attenuated, diffracted, and transmitted according to the material of the object. The acoustic material properties depend on the wavelength of sound and are impacted by characteristics such as surface roughness, density, and the presence of hidden resonant cavities. The materials in an environment have a strong influence on the sound heard by the listener.

When a sound wave is reflected from a flat rigid surface much larger than the wavelength, it is reflected in a mirror-like or specular manner. With specular reflections, the incoming wavefront is deflected in a direction that has the same angle relative to the surface normal. If the surface of an object has bumps or roughness that is of the same scale as the wavelength, the incident sound wave may be scattered rather than perfectly reflected. This kind of reflection is called a diffuse reflection. A perfectly diffuse surface would scatter the same amount of sound in all directions in the surface hemisphere. Many acoustic materials exhibit a mixture of specular and diffuse reflection at different frequencies. Reflections are responsible for producing echoes, delayed copies of the original sound. The first few reflections of sound from the environment are called the *early reflections*. When many successive high-order reflections occur, the sound becomes increasingly diffuse and smoothly-decaying *reverberation* is produced.

When the wavelength of sound is larger than features in the environment, diffraction effects become significant. Diffraction is a wave phenomenon where sound is scattered by features that are similar in scale to the wavelength. Diffraction is most evident at lower frequencies and is responsible for sound bending around corners or objects, as well as the scattering of diffuse reflections. Diffraction can enable a listener to hear a sound source even though the source may be visually occluded.

Sound that interacts with a surface in the environment may also have some of its energy transmitted into the material behind the surface. This process is known as sound transmission. During transmission, the sound wave undergoes a change of medium (e.g. from air to water), and this causes attenuation that varies with the sound frequency. Transmission allows sound to travel through walls, windows, and other non-rigid materials, then possibly exit back into the air. Transmission effects are an important consideration for noise control applications where it is useful to limit the amount of outside noise that enters a quiet building.

If sound changes medium during its propagation, it can also be refracted. Refraction is a bending of the pressure wave that occurs when the speed of sound changes. Refraction can occur in the atmosphere due to variation in air temperature and pressure at different elevations, or can occur due to transmission between materials with different speeds of sound. Refraction is responsible for the way that sound carries across a body of water at night. The water cools the air near the surface, decreasing the speed of sound and causing sound waves emitted upward from the source to bend down toward the ground on the other side of the water.

If sources, listeners, or objects in the environment move quickly, then Doppler shifting may occur. The Doppler effect is a phenomenon where the frequency of sound is changed due to compression or expansion of pressure waves caused by the moving objects. This effect is most commonly encountered when a vehicle drives past a stationary listener. As the vehicle approaches, its sound is raised in frequency because pressure waves are compressed ahead of the vehicle. Then, as the vehicle passes the listener the pitch of the vehicle's sound decreases as the pressure waves emitted from behind are expanded. The Doppler effect is more noticeable when sound sources move more quickly, i.e. a race car moving at 200 km/h will shift the frequency more than a slow-moving truck.

When sound arrives at the listener's position, it is affected by the listener's head and torso before it arrives in the listener's ear canals. This phenomenon, commonly referred to as *spatial sound*, produces variation in the level and delay of sound for different frequencies and directions. The sound heard by the left and right ears tends to be different, and the brain exploits those differences to perceive spatial information about the environment, including the location of the sound sources.

In comparison to visible light, a form of electromagnetic radiation, sound is characterized by its wide frequency range and long wavelength that is of the same scale as everyday objects. Therefore, diffraction effects are much more prominent for sound than light, which has too small a wavelength to diffract at human scales. Another difference between sound and light is that the phase and time delay of sound is perceptually important, whereas light travels so quickly that most phase differences are imperceptible to the human eye. The inclusion of phase and diffraction effects causes some significant differences between rendering algorithms designed for graphics and sound.

1.2 Motivation

Sound propagation has a wide variety of applications in diverse areas including the entertainment industry, architecture, engineering, noise control and teleconferencing.

Virtual reality (VR) has seen renewed interest in recent years as commodity head-mounted displays have become widely available. An important component of VR is the rendering of realistic sound that enhances the users's sense of presence and immersion in the virtual environment (Hendrix and Barfield, 1996). VR requires high-fidelity sound propagation to be rendered with low latency.

A related application to VR is training simulations. Training simulations are frequently used by the military and emergency services to prepare trainees for dangerous situations in a safe environment. In these simulations it is critical to generate audio that has realistic sound propagation in order to produce the most faithful simulation.

Video games are another area where sound propagation can significantly improve the experience. Like with VR, audio must be generated for the virtual environments found in games. While games may not always focus on generating the most realistic sonic environment, sound propagation can nevertheless be used to enhance atmosphere, immersion, and gameplay. For example, sound propagation can allow players in a multiplayer game to locate hidden adversaries by the sound of their footsteps.

In augmented reality (AR), sound propagation can be used to generate audio for virtual objects that are placed in real environments. The augmented audio can improve the users's sense that the virtual object exists physically if it is rendered with the sound propagation effects of the real room.

Teleconferencing is another application of sound propagation, where users communicate remotely using an audiovisual connection. Sound propagation can be used to render the remote audio in a way that makes the remote user seem as if they were in the local room.

In the movie, television, and music industries, reverberation is frequently added to sound recordings to enhance the plausibility. Sound propagation can reduce the amount of work involved for audio mixing engineers when trying to replicate a particular sonic environment using digital filters.

Another major application of sound propagation is in architectural acoustics, where it may be important to predict how a particular room or CAD model of an architectural space will sound before it is constructed. This is especially important for the design of concert halls and auditoriums that require certain acoustic characteristics. A significant focus of architectural acoustics is determining which materials (e.g. acoustic panels) should be used that produce the best sound for a particular application. Sound propagation also has applications in interactive walkthroughs, where it is useful in generating virtual walkthroughs of real spaces. Simulation of realistic sound in walkthroughs can enable a potential real estate buyer to avoid spaces that have unpleasant acoustics.

In city planning, it is important to simulate sound propagation so that noise can be effectively controlled in sensitive locations such as schools, hospitals, and residential areas. Sound propagation can be used in the placement of noise barriers in relation to loud places like airports and factories.

Within engineering applications, sound propagation can be used in the design process to achieve desired levels of acoustic performance, such as to reduce the interior noise of passenger cars and aircraft cabins or to reduce the exterior noise pollution generated by combustion engines.

1.3 Challenges

Due to the complexity of sound propagation phenomena, high-quality interactive simulation of sound propagation in dynamic environments remains a challenging problem. To achieve interactive performance, the simulation must be updated with a latency of less than about 100ms (Lindau, 2009). Many previous approaches have been proposed that simulate sound propagation phenomena using geometric techniques. The RAVEN system (Lentz et al., 2007) is a state of the art interactive geometric sound propagation system. It is mainly designed for indoor scenes and supports specular reflections, diffuse reflections, diffraction, and spatial sound. However, it takes a few seconds to compute a single simulation update, and therefore is not that suitable for highly interactive applications like virtual reality. The RESound system (Taylor et al., 2009) takes about $250 - 500$ms to model 3rd-order diffuse and specular reflections using ray and frustum tracing. More recent ray tracing methods have improved on these results by using guided ray tracing (Taylor et al., 2012), and can compute 3rd-order reflections and 1st-order diffraction in about 93ms for a few sources in scenes typical of interactive games. Other approximate methods have been proposed that estimate the reverberation in environments using heuristic algorithms (Antani and Manocha, 2013). These techniques can update the simulation in about 10ms, however they introduce numerous approximations and simplifications that can impact the plausibility of the results. In general, previous approaches are able to simulate a few orders of reflection and 1st-order diffraction for a few sound sources in a hundred milliseconds. More complex high-order reflections or diffraction can take a second or more to compute. As a result, there are still many challenges that must be addressed to enable low-latency interactive sound propagation:

Interactive performance: One of the largest problems faced by previous sound propagation techniques is achieving interactive performance in complex dynamic scenes with many sources. The fastest algorithms for computing sound propagation are based on geometric acoustics and ray tracing. However, the quality and performance of ray tracing depends strongly on the number of rays traced. Current sound ray tracing algorithms require a large number of rays to produce noise-free results. Roughly 10,000 - 50,000 rays must be emitted from each source (Lentz et al., 2007; Taylor et al., 2012), and those rays must be propagated for about 200 bounces to simulate late reverberation. This leads to about 2-10 million rays per source per simulation update. Recent advancements in ray tracing for graphics have enabled tens of millions of rays to be traced per second on the CPU (Wald et al., 2014). However, the performance on complex environments with many sound sources is not sufficient, since it may take hundreds of milliseconds for each sound source. As a result, it is difficult to apply ray tracing approaches directly to interactive applications due to the large update latency.

Acoustic material acquisition: Another significant challenge in computing sound propagation for virtual and augmented reality environments is that accurate acoustic material properties are necessary to compute plausible sound, yet are difficult to acquire. Some previous approaches manually assign a material for every surface from a database of measured material data. However, this process is time consuming and requires expert knowledge of how sound interacts with materials to produce accurate results. Automatic approaches have also been proposed that use optimization algorithms to assign materials properties (Monks et al., 2000; Christensen et al., 2014; Saksela et al., 2015), however they have very slow convergence or are limited to virtual scenes. Furthermore, the resulting simulations may not match known acoustic characteristics of real-world scenes due to inconsistencies between the measured data and actual scene materials. Mismatch between the sound simulation and real world can interfere with the subjective presence and realism of virtual sound sources placed within real-world environments.

Spatial sound: A drawback of current techniques for modeling spatial sound is that they focus on point sound sources and are inefficient for large sources that occupy an area or volume. As a result, current spatial sound techniques are not interactive when handling sources such as oceans, rivers and lakes. An additional challenge is that spatial sound cannot be rendered interactively for sound propagation effects. With sound propagation, sound arrives from various directions at different times. This greatly increases the complexity involved in rendering spatial sound and makes existing approaches non-interactive. The RAVEN system takes about 750ms to update the spatial sound filters during dynamic motion (Lentz et al., 2007). This large latency is experienced by the user as a delay or lag in the sound. The perception of latency is strongest when the listener is quickly rotating their head. This latency strongly detracts from the plausibility and interactivity of dynamic scenes.

1.4 Thesis Statement

Sound propagation can be efficiently rendered for interactive, dynamic, multi-source environments through the use of temporal coherence, automatic material classification, and low-latency spatial sound.

In this dissertation, we describe a collection of novel techniques that enable simulation and rendering of sound propagation in large, complex, multi-source environments at interactive rates. Our first goal is to improve the performance problems of existing interactive sound propagation algorithms by reusing computations executed on previous simulation time steps. Next, we demonstrate how acoustic materials can be automatically determined using a combination of visual and audio measurements of real-world environments. Finally, we propose spatial sound techniques that enable high-quality, low-latency directional audio to be efficiently computed for sound propagation. We have evaluated the impact of these improvements on a variety of benchmarks and shown that our approach enables efficient interactive simulation of sound propagation in complex, dynamic environments.

1.5 Main Results

In this dissertation, we present three approaches for interactive sound propagation in dynamic scenes. First we discuss how temporal coherence can be used to improve the interactive performance of sound propagation. Next, we propose an automatic acoustic material classification and optimization algorithm that is used to estimate the material properties that are present in real environments. Finally, we show how spatial sound can be efficiently rendered for sound propagation with low latency. Figure 1.1 illustrates how these ideas can be integrated into a complete sound propagation and rendering pipeline.

1.5.1 Temporal Coherence for Sound Propagation

While many previous techniques have been proposed for interactive sound propagation using geometric ray-tracing algorithms, a significant limiting factor is the number of rays that must be traced on each

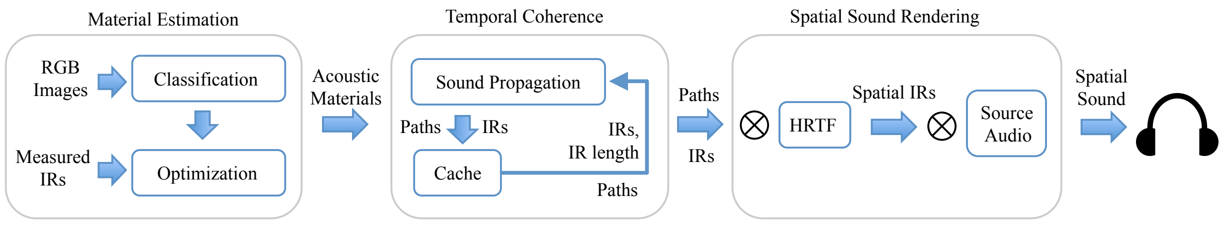


Figure 1.1: A high-level overview of our material estimation, sound propagation, and spatial sound rendering pipeline. We use RGB images of the environment along with measured impulse responses to automatically estimate the acoustic materials. Then, we use sound propagation and temporal coherence techniques to efficiently compute interactive sound propagation. The resulting propagation paths and impulse responses are used by the sound rendering module to compute the final audio using convolution with a spatial impulse response.

simulation update. As a result, previous methods are limited to either static precomputed scenes or only a few sound sources.

We propose a novel approach that reduces the work that must be performed on each time step by using sound propagation results from previous time steps. This cached information, including propagation paths and impulse responses, is stored in persistent data structures and is used to improve the simulation quality and reduce sampling noise caused by tracing too few rays. Our approach is based on the assumption that the sound field at the listener's position does not change quickly over time in most situations, e.g. there is *temporal coherence*. The main contributions of this technique include:

1. **Specular path cache:** A cache of specular early reflection paths from previous frames is maintained in order to reduce the computation needed to find specular paths on current and future frames.

2. **Diffuse path cache:** Diffuse early reflection paths are grouped together based on a surface subdivision and cached over several frames. A moving average filter uses paths from previous and current frames to estimate a diffuse sound field with less sampling noise.

3. **Impulse response cache:** A cache of the impulse response between each source and listener from the previous frame is used to improve the quality of high-order diffuse ray tracing with low computational and memory overhead.

4. **Adaptive impulse response length:** The listener's threshold of hearing is applied to the previous frame's impulse response in order to adaptively determine the length of the impulse response that should be computed on the current time step.

This technique has been tested on a variety of complex indoor and outdoor scenes, and has been shown to reduce the computation required, in terms of the number of rays, for interactive sound propagation by over an order of magnitude as compared to previous approaches based on ray tracing.

1.5.2 Acoustic Material Classification and Optimization

A significant issue with the extension of sound propagation to augmented reality and architectural applications is that the material properties of the real environment can be difficult to accurately acquire. Previous techniques have relied on manual assignment of materials, but this is time-consuming and may not always produce accurate results.

We propose a new automatic approach for estimation of acoustic materials in real environments. Audio and visual captures of the environment are used within a two-step process that involves using visual information to classify the material categories that are present, then optimizing those materials until the acoustic simulation in the virtual room matches the audio in the real room. The main contributions are summarized as follows:

1. **Material classification:** The visual appearance of the scene is captured in many color images which are the input to a convolutional neural network (CNN). The CNN has been trained to recognize common real-world material categories at each point in a color image. The material predictions are projected from each image into the scene and are then used to determine the final material categories for every surface primitive.

2. **Material optimization:** Given measured impulse responses from the real scene, an iterative optimization algorithm is used to improve the accuracy of the material estimates. On each iteration, a simulation is used to gather information about sound transport within the scene. Then, a least-squares formulation solves for the material absorption values that best match the measured impulse responses.

This approach has been applied to several real-world environments and its effectiveness has been both quantitatively and subjectively evaluated. With automatic material acquisition, realistic sound effects can be computed in real-world scenes where it would otherwise be difficult to generate plausible sound.

1.5.3 Low-Latency Spatial Sound for Sound Propagation

The computation of accurate spatial sound is a significant bottleneck for interactive sound rendering when applied to the complexity of realistic sound propagation. In this case, sound arrives at the listener from many directions at different delay times, and each sound arrival must be auralized with the directional filtering of the listener's head geometry.

We present a technique for efficient rendering of spatial sound with low latency using the spherical harmonic (SH) basis functions. Our approach represents the sound field at the listener's position in the SH domain, and then takes advantage of SH orthonormality to efficiently construct spatial sound filters for area and volume sources as well as sound propagation impulse responses. The primary contributions of our technique include:

1. **Area and volume sources:** The projection of sound pressure arriving at the listener's position from all directions is used to compute the spatial sound filter for sources that emit sound over a large area or volume.

2. **Spatial impulse response construction:** A perceptual metric is used to adaptively determine the spherical harmonic order to use for each partition of a sound propagation impulse response.

We have implemented this approach within the Unity™game engine and evaluated the performance and subjective quality of the results with user studies. A speedup of at least an order of magnitude is achieved in each case over the previous technique, enabling high-quality spatial sound to be computed with low latency for interactive sound propagation.

1.6 Organization

The remainder of the dissertation is organized according to the following structure:

- In **Chapter 2**, we introduce relevant background and previous work on sound propagation, acoustic materials, audio rendering, and spatial sound.

- **Chapter 3** describes how temporal coherence can be used to improve the performance of interactive sound propagation in dynamic scenes.

- In **Chapter 4**, a technique for automatic acquisition of acoustic materials in real world scenes is presented, along with comparisons to real-world measurements.
- **Chapter 5** introduces techniques for efficiently rendering spatial sound filters for area and volume sound sources, as well as a method for perceptual spatial impulse response construction.
- **Chapter 6** concludes the dissertation and discusses potential avenues for future work.

CHAPTER 2: BACKGROUND

2.1 Sound Propagation

When sound propagation is computed for a virtual environment, the goal is to determine the how the sound emitted by each source is affected by the environment before it is received at the listener. This can be modeled as a filter whose impulse response (IR) specifies how sound is delayed and attenuated due to sound propagation. Sound propagation algorithms can be divided into two main classes: those that numerically solve the acoustic wave equation, the so-called *wave-based* methods, and those that use geometric algorithms to approximate sound propagation phenomena, the *geometric* methods.

2.1.1 Wave-based Sound Propagation

Wave-based sound propagation algorithms are both the most accurate and computationally expensive methods. Time-domain solvers divide the simulation domain into cubic or rectangular volume partitions that have an analytical solution to the wave equation, then compute how sound waves propagate within and between the partitions. These include Finite Difference Time Domain (FDTD) (Savioja, 2010) and Adaptive Rectangular Decomposition (ARD) (Raghuvanshi et al., 2009). Other approaches such as the Boundary Element Method (BEM) solve the wave equation in frequency domain given the boundary conditions on every surface in the scene (Ciskowski and Brebbia, 1991). While wave-based methods are widely used in scientific and engineering applications, the computation time and memory required scales very poorly with the maximum frequency and size of the simulation domain. Therefore they are usually limited to offline simulation. On the other hand, precomputation and compression approaches have been proposed that enable interactive wave-based sound propagation with a dynamic source and listener in static scenes (Raghuvanshi and Snyder, 2014).

2.1.2 Geometric Sound Propagation

Geometric sound propagation techniques are based on the simplifying assumption that the surface primitives in the environment are much larger than the wavelength of sound. As a result they can more easily simulate dynamic scenes at interactive rates. While this assumption is valid for high frequencies, geometric algorithms are inherently less accurate at low frequencies where wave phenomena like diffraction must be modeled explicitly.

2.1.2.1 Specular Reflections

Many algorithms have been proposed for the computation of specular reflections. In the image source method (Allen and Berkley, 1979; Borish, 1984), point sound sources are recursively reflected over every surface primitive to form a tree of possible reflection paths, then the valid unoccluded paths are determined by tracing rays from the listener position back to the source. The complexity of the basic image source method can be improved by using ray tracing to sample the most likely reflection paths (Ondet and Barbry, 1989). In beam tracing (Funkhouser et al., 1998) and frustum tracing (Chandak et al., 2008), rectangular beams or frusta are propagated from the source and recursively reflected to build a tree of possible paths that are validated using ray tracing. A significant drawback of these techniques is that they are limited to specular reflections and cannot model the later parts of the impulse response where the sound becomes more diffuse (Lentz et al., 2007).

2.1.2.2 Diffuse Reflections

The most common techniques for computing diffuse reflections are based on Monte Carlo path tracing, a probabilistic technique for solving complex integral equations (Krokstad et al., 1968; Vorländer, 1989; Embrechts, 2000). In path tracing, rays that each carry a fraction of the total sound energy are randomly emitted from the source. The rays are reflected throughout the environment until an intersection with the listener is detected, thereby generating a sound propagation path that is accumulated to the impulse response. The convergence of path tracing can be improved by direct sound sampling techniques such as *diffuse rain* (Schröder et al., 2007). Path tracing is often computed in separate frequency bands to model frequency-dependent scattering and attenuation. Radiosity may also be used for the computation of diffuse sound (Nosal et al., 2004), but cannot handle dynamic scenes.

2.1.2.3 Diffraction Modeling

Diffraction can be approximated within the geometric sound propagation framework using specialized diffraction techniques. The simplest methods are based on single-point diffraction over infinite edges and include the Geometric Theory of Diffraction (GTD) (Keller, 1962) and the Uniform Theory of Diffraction (UTD) (Kouyoumjian and Pathak, 1974). While computationally efficient to evaluate, these techniques become much less accurate when applied to complex environments with many small edges. The more expensive Biot-Tolstoy-Medwin (BTM) method (Svensson et al., 1999) instead computes a line integral over diffraction edges and can handle edges of any length with higher accuracy. Another approach for geometric diffraction modeling is based on the Heisenberg uncertainty principle (Stephenson, 2010) and it can be easily integrated within a path tracing framework.

2.2 Acoustic Materials

The acoustic materials that are present in an environment have a strong influence on the resulting sound propagation. One of the most important material properties is the absorption coefficient, $\alpha \in [0, 1]$. The absorption coefficient describes the fraction of incident sound pressure that is absorbed when a sound wave is reflected by a surface. α is usually a function of both frequency and the angle between the incident direction and the surface normal. It can be measured according to ISO 354 (ISO, 2003) where a sheet of the material to be measured is a placed in a reverberation chamber, then the absorption is derived from the difference in reverb time using the Sabine equation (Sabine, 1922).

The scattering behavior of a material is most commonly described by the frequency-dependent scattering coefficient $s \in [0, 1]$. s represents the fraction of incident sound that is reflected diffusely (e.g. with a Lambertian distribution) and is correlated to the roughness of the surface (Christensen and Rindel, 2005). More complex scattering can be represented using bidirectional reflection distribution functions (BRDFs) that describe the outgoing energy distribution for a given incident direction and frequency (Mückl and Dachsbacher, 2014), though acoustic BRDFs are difficult to measure in practice.

Many approaches have been proposed that determine the acoustic materials of an environment using optimization techniques. Monks et al. (Monks et al., 2000) present a method for optimizing the placement of absorption in architectural acoustics using steepest descent and simulated annealing given target values for various impulse response metrics. Genetic algorithms have also been used to estimate acoustic materials so that the resulting simulations match real-world measurements, but the rate of convergence is very slow (Christensen et al., 2014). (Saksela et al., 2015) use least squares regression in combination with beam tracing to optimize surface absorption coefficients for virtual rooms. The impedance of acoustic materials in a real scene can be estimated for individual frequencies using acoustic measurements of pure tones and the inverse boundary element method (Nava, 2006).

2.3 Sound Rendering

The earliest techniques for rendering sound propagation are based on artificial reverberators, recursive feedback-delay filters that mimic the rate of decay of natural late reverberation in indoor rooms (Schroeder, 1961). The realism of artificial reverberators can be improved by rendering early reflections using additional delay taps (Gardner, 2002), or by simulating frequency-dependent sound absorption using a low pass filter in the feedback loop (Schroeder, 1961). The reverberation parameters can also be initialized by analysis of the impulse response between the source and listener (Taylor et al., 2009). While these techniques are computationally efficient, they are unable to simulate time-varying directional reverberation or outdoor environments.

More sophisticated sound rendering methods use convolution of the source audio with the impulse response to generate realistic sound. Naïve time-domain convolution is expensive for long impulse responses, but IR partitioning schemes and frequency domain convolution can be used to improve both the latency and performance (Gardner, 1994). Convolution can reproduce most sound propagation phenomena except for Doppler shifting.

To render the pitch shifting inherent to the Doppler effect, fractional delay interpolation can be used on direct and early reflection paths. However, it is too expensive to render Doppler shifting for all sound propagation paths for interactive applications. Delay interpolation involves reading from a circular delay buffer at a rate that depends on the relative velocity between the source and listener along the propagation path, thereby either compressing or expanding the sound waves along the time axis (Strauss, 1998).

2.3.1 Spatial Sound

A perceptually important aspect of sound rendering is the generation of spatial sound such that the listener has the impression of a 3D directional sound field. The simplest method is vector-based amplitude panning (VBAP) (Pulkki, 1997), where the vector to the sound source is used to compute the amplitude of each channel in a 3D array of loudspeakers. While computationally efficient, VBAP does not model phenomena like the different times of sound arrival at each ear or frequency-dependent scattering from the head and torso. The directional behavior of the listener is best described by the head-related transfer function (HRTF). The HRTF is a function of either time or frequency on the spherical domain. By interpolating an HRTF measured for a listener and convolving the interpolated HRTF filter with the audio for the source, the listener is given the impression that the source is in a particular direction. To generate HRTF-based spatial sound for sound propagation, the HRTF is convolved with the impulse response to generate a spatial impulse response (SIR), then the SIR is convolved with the source audio. Due to the expense of convolving the HRTF with the IR, current methods either only apply the HRTF to important direct and early reflection paths, or cluster paths based on direction to reduce the number of HRTF interpolations (Lentz et al., 2007). However these methods introduce error and are still too slow for dynamic interactive simulations. As a result, it remains computationally expensive to combine spatial sound with sound propagation.

CHAPTER 3: TEMPORAL COHERENCE FOR SOUND PROPAGATION [1]

3.1 Introduction

In simulations of interactive sound propagation, it is common to update the simulation state in a series of discrete time steps that are referred to as *frames*. On each frame, a new collection of impulse responses between each source and listener are computed and used for auralization. While traditional geometric acoustics algorithms based on ray tracing are among the fastest ways to compute these impulse responses, the quality and performance of ray tracing depends strongly on the number of rays traced and how many orders of reflection are computed. Sound propagation approaches based on Monte Carlo path tracing involve generating many random rays to sample sound propagation paths within the scene. Those paths are then used to estimate the sound intensity received by the listener. The more rays that are traced, the more accurate this estimation will be, but at increased computational expense. If not enough rays are traced, the space of propagation paths will not be effectively sampled, resulting in noise in the computed impulse response. This noise is most audible in interactive sound propagation where the impulse response and rendered audio changes noticeably each time it is updated. Rendering directional spatial sound exacerbates this effect, producing sound that seems to move around the listener as different propagation paths are detected on each simulation update. As a result, the number of rays that are required for artifact-free audio can be very large. In addition, it is not known *a priori* how many secondary ray bounces should be computed for each primary ray, as this depends on factors such as the scene geometry, the position and loudness of the source, and the acoustic materials present in the scene. If not enough bounces are computed, it can truncate the end of the impulse response, while if too many bounces are computed, the unnecessary computation may be wasted. If these ray parameters are not carefully chosen, it can either negatively affect the sound quality, or lead to non-interactive performance for complex scenes.

In this chapter, we present techniques that are based on the observation that the sound at the listener does not change very quickly in most scenes, i.e. there is *temporal coherence*. Our algorithms use information

[1]Much of this chapter appeared previously Schissler and Manocha (2011); Schissler et al. (2014); Schissler and Manocha (2016a)

computed on previous simulation frames, such as sound propagation paths and impulse responses, to reduce the number of rays and ray bounces that are needed on each frame for sufficient sound quality. Similar ideas have been proposed in the graphics literature in order to reuse shading results from previous frames or to incorporate temporal antialiasing (Scherzer et al., 2011). First, we focus on how temporal coherence can be utilized in the computation of specular early reflections (Section 3.2.1). A data structure known as the *specular path cache* is used to cache specular reflection paths from previous frames, thereby reducing the cost involved in finding paths on future frames, allowing fewer rays to be traced, and improving performance. Next, we propose an approach for diffuse reflections that uses a *diffuse path cache* to utilize ray tracing from previous frames (Section 3.2.2). A moving average filters the sound intensity for each diffuse reflection path over many frames. This increases the accuracy of Monte Carlo path tracing by taking more rays into account. In Section 3.2.3, we present another temporal coherence technique that uses a cached copy of the previous impulse response, the *impulse response cache*, in order to filter the path tracing IR using exponential smoothing (Brown, 1956). This approach is notable in that it can be implemented very efficiently and enables fast computation of high-order reflections. Finally, we describe a method for adaptively determining the length of the impulse response using the IR from the previous frame and a perceptually-based threshold (Section 3.2.4). The IR length is used to determine how many secondary ray bounces should be computed on the next frame without truncating the IR or performing unnecessary computation.

We have evaluated the subjective and performance impacts of these approaches on a variety of complex scenes and demonstrate that temporal coherence can enable dynamic sound propagation effects to be computed at interactive rates.

3.2 Temporal Coherence for Sound Propagation

In this section we describe temporal coherence techniques that can be used to improve the quality and performance of interactive sound propagation. An overview of our approach is shown in Figure 3.1. We utilize several data structures to cache information that improves the results for various components of sound propagation. These include the *specular path cache* (Section 3.2.1), the *diffuse path cache* (Section 3.2.2), and the *impulse response cache* (Section 3.2.3). Using these ideas, we obtain significant benefit over previous interactive sound propagation approaches.

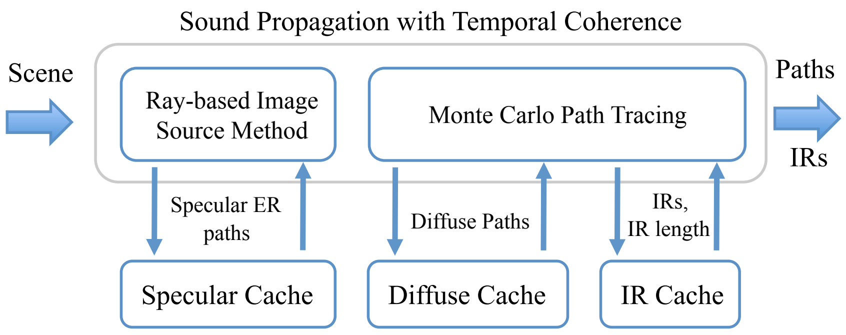


Figure 3.1: A high-level overview of our sound propagation technique that uses temporal coherence. We cache specular reflection paths, diffuse reflection paths and impulse responses in order to improve the sound quality for interactive applications.

3.2.1 Specular Path Cache

The early reflections (ER) are a perceptually important component of sound propagation that correspond to the first several orders of reflection and that consist of mostly specular sound energy (Lentz et al., 2007). A widely-used algorithm for computation of specular reflections is the image source method (Allen and Berkley, 1979; Borish, 1984). The image source method as originally described has a complexity of $O(N_T^{d_{max}})$, where N_T is the number of triangles in the environment, and d_{max} is the maximum reflection order. As a result, various acceleration techniques have been proposed such as beam tracing (Funkhouser et al., 1998) and frustum tracing (Chandak et al., 2008) that use volume intersection queries to reduce the number of possible reflection paths that must be checked for validity. The ray-based image source method (RISM) (Ondet and Barbry, 1989) instead uses stochastic ray tracing to efficiently sample possible paths. Unlike beam and frustum tracing, the performance and quality of the ray-based image source method is largely dependent on the number of primary rays that are emitted from the source or listener. If not enough rays are traced, some propagation paths may not be explored, and if different random rays are traced on each frame it can introduce unnatural variation in the sound from frame to frame. However, if too many rays are traced, computation is wasted on redundant path validation.

In this section we propose an algorithm that is an extension to the ray-based image source method and utilizes temporal coherence to reduce the number of rays traced while maintaining similar sound quality. A primary component of our algorithm is a data structure called the *specular path cache* (SPC). The specular path cache is used within the RISM to reduce the number of rays needed and to reduce unnatural variation in the sound over time. The SPC stores a mapping from previously validated specular reflection paths to information about the paths, namely whether or not the path is valid. It is implemented as a hash table so that the contents can be randomly accessed in $O(1)$ time. On each sound propagation frame, the RISM is used to generate and validate possible specular reflection paths. Before each path is validated, the SPC is accessed to see if the path has been already explored by ray tracing on the current or previous frames. If so, computation can be saved because the path has already been checked for validity. If the path is not in the cache, the path is checked for validity, and the results of that test are stored in the cache. This enables specular reflections to be computed more efficiently and with higher quality than existing approaches.

Specular Path Identifier: Each specular reflection path can be uniquely identified by a path identifier, K, which represents a sequence of reflections between a given source and listener. For example, the path identifier for a 2nd order reflection could be $K = \{S_k, T_{j_1}, T_{j_2}, L_l\}$. This corresponds to a reflection path emitted by sound source S_k that reflects off triangles T_{j_1} and T_{j_2}, before being recieved at listener L_l. The path identifier is used as the key to access the SPC. If the RISM is implemented as a depth-first ray traversal, the path identifier can be efficiently incrementally constructed with each successive ray reflection.

Algorithm Details: We summarize our specular ray tracing algorithm using the SPC in Algorithm 1. The procedure begins by emitting N_r uniformly-distributed random rays from each sound source in the scene. These rays are propagated through the scene up to maximum specular reflection order d_{max}, and during the propagation for each primary ray we incrementally build the path identifier K, as well as a sequence of image sources, *IS*. Each ray is intersected with the scene geometry and if there is no intersection, the ray is terminated. Otherwise, we get the intersected triangle and reflect the previous image source across the triangle's plane. We also specularly reflect the incoming ray to generate the next outgoing ray. Then, for each listener in the scene we check to see if the current path identifier is contained in the SPC. If so, then the path is skipped because it has been previously validated on the current frame. If the path is not already in the SPC, we validate the path using the standard image source method. The result of that validation is stored in the SPC so that it can be reused for subsequent rays and time steps. This algorithm proceeds until all rays have been traced for all sources. At the end of the ray tracing process, the SPC contains a mapping from every specular reflection path that was explored to the valid status for the path (either *valid* or *invalid*). Paths that are *valid* contribute to the impulse responses between their respective sources and listeners.

Algorithm 1 Ray-based Image Source Method with the Specular Path Cache (SPC).
This algorithm describes how the specular path cache can be integrated with the ray-based image source method. We show the algorithm for forward ray tracing (from sources), though it can be more efficient to trace rays from the listener in multi-source scenes because the primary rays are reused for all sources.

```
1:  procedure SPECULAR RAY TRACING
2:      $N_r \leftarrow$ number of primary rays
3:      $N_S \leftarrow$ number of sound sources
4:      $N_L \leftarrow$ number of listeners
5:      $d_{max} \leftarrow$ maximum specular reflection order
6:      SPC $\leftarrow$ previously validated paths
7:      for $k = 1...N_S$ do
8:          $K \leftarrow \{S_k\}$
9:          for $i = 1...N_r$ do
10:             ray $\leftarrow$ uniform random ray starting at $S_k$
11:             IS $\leftarrow$ {position of $S_k$}
12:             for $d = 1...d_{max}$ do
13:                 if $\neg$IntersectScene(ray) then
14:                     break
15:                 end if
16:                 $T_{j_d} \leftarrow$ intersected triangle
17:                 IS $\leftarrow \{IS,$ ReflectPoint(IS$_d, T_{j_d}$)$\}$
18:                 ray $\leftarrow$ ReflectRay(ray$, T_{j_d}$)
19:                 $K \leftarrow \{K, T_{j_d}\}$
20:                 for $l = 1...N_L$ do
21:                     $K_l \leftarrow \{K, L_l\}$
22:                     if $K_l \notin$ SPC then
23:                         if Validate($K_l$, IS) then
24:                             SPC $\leftarrow$ SPC $\cup \{K_l \mapsto$ valid$\}$
25:                         else
26:                             SPC $\leftarrow$ SPC $\cup \{K_l \mapsto$ invalid$\}$
27:                         end if
28:                     end if
29:                 end for
30:             end for
31:         end for
32:     end for
33: end procedure
```

At the beginning of the next simulation frame, the contents of the SPC must be revalidated in case any source, listener, or scene geometry moved during the time interval. This process is described by Algorithm 2. We begin by iterating over the paths that are stored in the cache. For each path, we check to see if the path status in the cache is *invalid*. If so, the path is removed from the SPC. If the path is *valid*, the path is then revalidated based on the current positions of the source, listener, and triangles. To validate the path, we first generate a new sequence of image sources, *IS*, by recursively reflecting the new source position over the path's triangles. Then, the path is validated using the standard image source method validation procedure. The result is then used to update the path status within the SPC to either *valid* or *invalid*. After the entire contents of the SPC are revalidated, the SPC contains only paths that were *valid* on previous frame(s). The SPC can then be used again in Algorithm 1 to detect more specular reflection paths and add them to the cache.

Algorithm 2 Specular Path Cache Validation.
This algorithm describes how the contents of the specular path cache are validated at the beginning of each sound propagation frame before any rays are traced. The paths that were invalid on the previous frame are removed from the cache, while the valid paths are revalidated and stored again in the cache.

```
procedure SPC VALIDATION
    SPC $\leftarrow$ validated paths from previous frame
    for $K \in$ SPC do
        if PreviouslyValid$(K, \text{SPC})$ then
            $S_k \leftarrow$ starting source of path $K$
            $L_l \leftarrow$ ending listener of path $K$
            $IS \leftarrow \{$position of $S_k\}$
            $d_{max} \leftarrow$ number of reflections in path $K$
            for $d = 1...d_{max}$ do
                $T_{j_d} \leftarrow$ triangle in $K$ at reflection order $d$
                $IS \leftarrow \{IS, \text{ReflectPoint}(IS_d, T_{j_d})\}$
            end for
            if Validate$(K, IS)$ then
                SPC $\leftarrow$ SPC $\cup \{K \mapsto$ valid$\}$
            else
                SPC $\leftarrow$ SPC $\cup \{K \mapsto$ invalid$\}$
            end if
        else
            Remove path $K$ from the SPC
        end if
    end for
end procedure
```

By using the SPC, we improve the ray-based image source method in a few ways. First, the cache ensures that the sound does not change in an unnatural manner over time. Unlike the basic RISM, once a path is found and added to the cache, the path will always be found on subsequent frames until it is invalidated by occlusion. This means that in a static scene, our approach will continue to add paths to the cache until the space of specular reflection paths is completely explored. In a dynamic scene, our approach does not introduce any additional error because paths that become invalidated are immediately removed from the cache during the validation procedure at the beginning of each frame (Algorithm 2).

Another way that we improve over the RISM is that the SPC allows for much fewer rays to be traced on each frame, thereby significantly improving the performance and overall latency of the sound propagation without impacting the quality. For example, with the standard RISM if we emit $N_r = 10,000$ primary rays from each source it takes $t_{10\text{k}}$ milliseconds to compute sound propagation paths. With the addition of the SPC, we can emit only $N_r = 1,000$ primary rays and it will take $t_{1\text{k}}$ milliseconds. Since the performance is dominated by ray tracing and the SPC itself does not add much additional computation, $10 \times t_{1\text{k}} \approx t_{10\text{k}}$. Because the SPC allows the ray tracer to incrementally determine the propagation paths over many frames, rather than compute all paths on each frame, the results at time $t = t_{10\text{k}}$ will be the same for both methods because the total number of rays traced is the same $(10,000)$. With our approach based on the specular path cache, we can trace fewer rays on each frame, and thereby achieve a much lower overall latency and greater interactivity while maintaining equal or better quality to the standard ray-based image source method.

3.2.2 Diffuse Path Cache

A diffuse reflection occurs when sound incident on a rough surface is scattered in non-specular directions. While specular reflections can be easily computed in closed-form using the image source method, diffuse reflections typically require a numerical approach. A commonly used technique is Monte Carlo path tracing (Embrechts, 2000). In path tracing, many random rays are emitted from each source and/or listener, then scattered through the scene up to maximum reflection order d_{max}, with the goal of finding paths between the source and listener. Since diffuse path tracing is a Monte-Carlo method, it requires a large number of ray samples to generate accurate results. If not enough rays are traced, it results in impulse responses with lots of noise. This noise is audible as unnatural variation in the sound from frame to frame. It is most noticeable with spatial sound rendering because the sound in a static scene seems to move around the listener and arrive from different directions over time. One solution is to trace the same rays on each frame, but this biases the computation and does nothing to improve the performance of path tracing. Therefore, current techniques

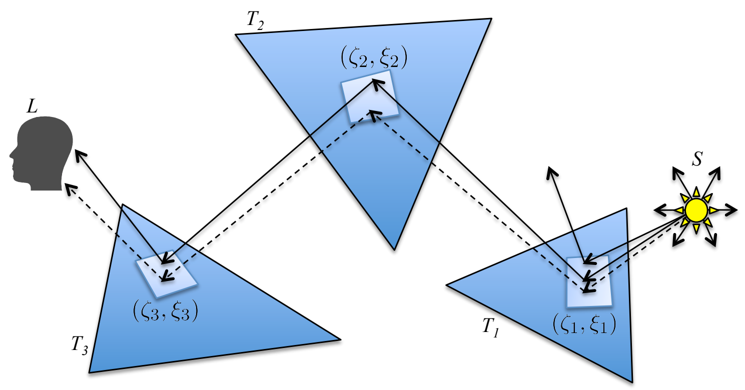


Figure 3.2: In our diffuse cache algorithm, rays leave the sound source S, hit a sequence of surface patches $\{T_1(\zeta_1, \xi_1), T_2(\zeta_2, \xi_2), T_3(\zeta_3, \xi_3)\}$, then hit the listener L. Rays with dashed paths are from previous frames, while rays with solid paths are from the current frame. Our technique groups these coherent rays together because they hit the same sequence of surface patches. The sound contribution at the listener is averaged over the time period τ, using rays from the previous and current frames, resulting in less noise in the impulse response compared to traditional path tracing approaches.

for diffuse reflections can only compute $1 - 3$ orders of reflections interactively and they must trace a large number of rays (Lentz et al., 2007; Taylor et al., 2009).

To address these issues, we propose an algorithm that takes advantage of temporal coherence to accelerate the computation of diffuse reflections for interactive sound propagation. Our approach integrates with traditional Monte Carlo path tracing and uses a cache of diffuse reflection paths from the current and previous frames to reduce the variance in the Monte Carlo estimation. This data structure is known as the *diffuse path cache* (DPC). The DPC contains a mapping from path identifiers to information describing the sound intensity that travels along each path. Each time that a diffuse reflection path is detected during path tracing, we use the path to update the contents of the DPC. We use a radiosity-like subdivision of the surface geometry into patches in order to group together similar diffuse reflection paths in the DPC. The DPC entries use a moving average of sound intensity over multiple time steps to produce a better estimate of the actual sound intensity received at the listener. This process is illustrated in Figure 3.2. As a result, our algorithm is able to incrementally compute the diffuse sound field over many frames by reusing rays and therefore requires much fewer rays to achieve the same sound quality as traditional path tracing.

Patch Subdivision: As part of a preprocessing step, we subdivide the triangles in the scene into a set of surface patches. This operation can also be done efficiently at runtime if the scene geometry deforms. The patch subdivision is used to group together similar diffuse reflection paths within the DPC. We ensure that each patch in the subdivision is roughly the same size and meets minimum spatial size criteria. We use Barycentric coordinates to partition each triangle in the input scene into a grid of quadrilateral and triangular patches. Patches are arranged as a 2-dimensional grid of entries with indices (ζ, ξ), as shown in Figure 3.3. We do not store these patches explicitly; instead we use the Barycentric coordinates of each ray-triangle intersection, along with precomputed information about the triangle, to determine which surface patch contains the intersection point at runtime. This formulation requires only a few extra bytes per triangle.

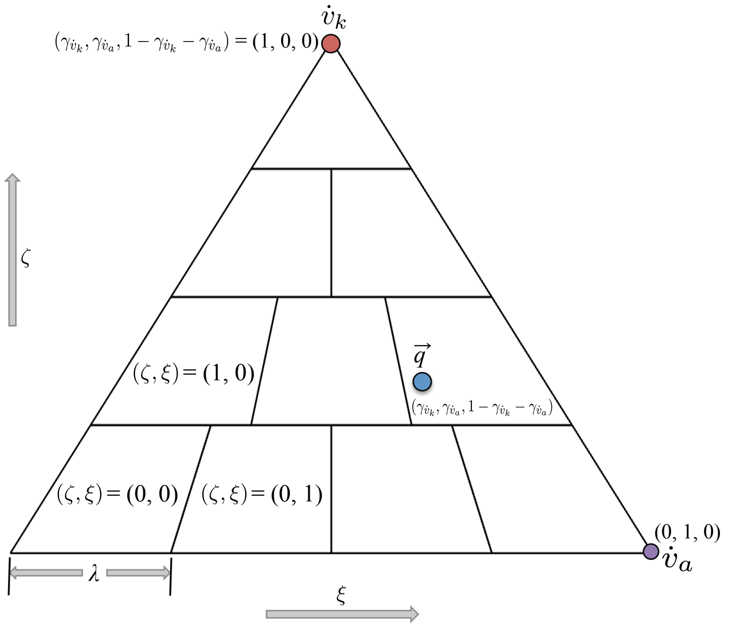
In order to precompute the subdivision for a given triangle T_i, we select a vertex $\dot{v}_k$ of T_i as the *key* vertex for that triangle: the vertex that is incident to the longest altitude of T_i. The length of the altitude from $\dot{v}_k$, h_k, is used to determine the number of rows in the subdivision $N_\zeta = \lceil h_k/\lambda \rceil$, where λ is a parameter used to govern the resolution of the subdivision. In addition, the number of columns for the largest row is $N_\xi = \lceil e_k/\lambda \rceil$ where e_k is the length of the edge opposite $\dot{v}_k$. The number of columns for the ζth row, N_ξ^ζ, is determined by

$$N_\xi^\zeta = \left\lceil \frac{N_\zeta - \zeta}{N_\zeta} N_\xi \right\rceil . \tag{3.1}$$

In order to determine this subdivision at runtime, we only store the values of N_ζ, N_ξ, and the index of key vertex $\dot{v}_k$ for each triangle. The choice of subdivision size λ determines the size of the patches and accuracy of the approach, as in radiosity-based algorithms. In general, λ should be chosen such that the incident sound intensity doesn't change too much across adjacent patches.

Diffuse Path Identifier: After the triangles in the environment have been subidivided into patches, every diffuse path can be identified by the sequence of surface patches that it involves. The identifier for the jth diffuse path is given by $K_i = \{S_k, T_{i_1}(\zeta_{i_1}, \xi_{i_1}), ..., T_{i_d}(\zeta_{i_d}, \xi_{i_d}), L_l\}$, where d is the number of diffuse reflections that occurred along the path, S_k is the sound source, and L_l is the listener.

Diffuse Path Cache: The diffuse path cache is implemented as a hash table that contains mappings from path identifiers K_i to information that is used to incrementally compute the sound intensity for each cache entry. The ith cache entry, corresponding to a group of ray paths, stores the values $(\eta_i, \mu_i, \hat{I}_i(\omega), \hat{r}_i)$. η_i is the number of rays along this entry's path that have reached the listener. μ_i is the total number of rays emitted from the source for all frames, while this entry was in the cache. $\hat{I}_i(\omega) = \sum I(\omega)$ is the sum of the total frequency-dependent sound intensity $I(\omega)$ for all rays that have traveled the path for this entry. $\hat{r}_i = \sum r$ is



$\dot{v}_k$
$(\gamma_{\dot{v}_k}, \gamma_{\dot{v}_a}, 1-\gamma_{\dot{v}_k}-\gamma_{\dot{v}_a}) = (1, 0, 0)$
ζ
$\vec{q}$
$(\zeta,\xi) = (1, 0)$
$(\gamma_{\dot{v}_k}, \gamma_{\dot{v}_a}, 1-\gamma_{\dot{v}_k}-\gamma_{\dot{v}_a})$
$(\zeta,\xi) = (0, 0)$
$(\zeta,\xi) = (0, 1)$
(0, 1, 0)
$\dot{v}_a$
λ
ξ

Figure 3.3: An example triangle subdivision. The triangle is subdivided into an array of indexed patches (ζ, ξ) based on the subdivision resolution λ. We compute the ray intersection point $\vec{q}$ with Barycentric coordinates $(\gamma_{\dot{v}_k}, \gamma_{\dot{v}_a}, 1 - \gamma_{\dot{v}_k} - \gamma_{\dot{v}_a})$, (e.g. $(\zeta, \xi) = (1, 3)$).

the sum of the path lengths r for all rays that have hit this sequence of surface patches while the entry was in the cache. From these values, the average incident sound source intensity received at the listener, I_i, as a fraction of the total emitted energy can be computed by:

$$\bar{I}_i(\omega) = \frac{\eta_i}{\mu_i} \frac{\hat{I}_i(\omega)}{\eta_i}. \tag{3.2}$$

In this relation, $\frac{\hat{I}_i(\omega)}{\eta_i}$ computes the moving average of the total sound intensity for all rays that travel path i. The ratio of $\frac{\eta_i}{\mu_i}$ has the effect of dividing the output intensity by the number of frames that have been computed while the entry was in the cache. To compute the average path length $\bar{r}$ for a cache entry, we use:

$$\bar{r} = \frac{\hat{r}}{\eta}. \tag{3.3}$$

This average path length is divided by the speed of sound c in the propagation medium to determine the average delay time for each path.

Path Tracing with the Diffuse Path Cache: At the beginning of each simulation step, we trace uniformly-distributed random rays from each sound source and diffusely reflect those rays through the scene according to the material BRDFs up to a maximum reflection order d_{max}, as in traditional path tracing. For a ray-triangle intersection at reflection order d and ray i, we first find the surface patch, $T_{i_d}(\zeta_{i_d}, \xi_{i_d})$, for the intersection point $\vec{q}_{i_d}$ on triangle T_{i_d}. We compute the Barycentric coordinates $(\gamma_0, \gamma_1, \gamma_2)$ of $\vec{q}_{i_d}$ with respect to triangle T_{i_d}. Next, we choose two of the three components of the Barycentric coordinates $(\gamma_{\dot{v}_k}, \gamma_{\dot{v}_a})$ from the set $(\gamma_0, \gamma_1, \gamma_2)$ in order to define the subdivision axes. $\gamma_{\dot{v}_k}$ is the component corresponding to the key vertex $\dot{v}_k$, and $\gamma_{\dot{v}_a}$ is the component for the vertex $\dot{v}_a$ that is to the left of $\dot{v}_k$ on triangle T_{i_d}. Given $\gamma_{\dot{v}_k}$ and $\gamma_{\dot{v}_a}$, we can compute the row and column indices (ζ_{i_d}, ξ_{i_d}) for the surface patch containing $\vec{q}_{i_d}$, as shown in Figure 3.3: $\zeta_{i_d} = \lfloor \gamma_{\dot{v}_k} \cdot N_\zeta \rfloor$, $\xi_{i_d} = \left\lfloor \gamma_{\dot{v}_a} \cdot N_\xi^\zeta \right\rfloor$. This patch $T_{i_d}(\zeta_{i_d}, \xi_{i_d})$ is added to the patch identifier K_i for the ray that is currently being propagated.

When the ray is reflected, the outgoing ray is tested to see if it intersects each listener. If so, we form the path identifier $K_i = \{S_k, T_{i_1}(\zeta_{i_1}, \xi_{i_1}), ..., T_{i_d}(\zeta_{i_d}, \xi_{i_d}), L_l\}$ based on the reflections that occurred along this path from the source S_k to listener L_l. The path identifier is then used to access the diffuse cache in $O(1)$ time. If there was an existing cache entry for that specific patch sequence, the entry is updated with the contribution for that ray:

$$\eta_i' = \eta_i + 1; \quad \hat{I}_i'(\omega) = \hat{I}_i(\omega) + I_i(\omega); \quad \hat{r}_i' = \hat{r}_i + r_i. \tag{3.4}$$

If there is no entry corresponding to this reflection path, a new entry is inserted into the cache for identifier K_i and the corresponding parameters are set as $\eta_i = 1$, $\mu_i = 0$, $\hat{I}_i(\omega) = I_i(\omega)$, $\hat{r}_i = r_i$. After all rays have been traced from the source and the cache entries updated for rays that reached the listener, the cache contains entries that correspond to the accumulated contribution of groups of rays that have traveled along similar paths to the listener during the current frame or previous frames.

Impulse Response Computation: Next, we compute the final impulse response for each source and listener pair from the cache by iterating through all entries and generating a delayed impulse for each entry that is added to the output intensity impulse response, $I_{\bar{\omega}}(t)$. During this process, the value of μ_i is increased for each entry by the total number of rays emitted from the source during this frame. We use equation (3.2) to compute the incident intensity $I_i(\omega)$ for this cache entry. The delay time t_i for the ith cache entry is computed using the average path length from equation (3.3) and the speed of sound. Finally, the intensity of the cache entry, $I_i(\omega)$ is added to the output impulse response at delay time t_i.

Cache Response Time: In order to avoid storing reflectance data that is no longer accurate for the current scene configuration, we also bound the maximum age in seconds of the data stored in the cache. Any cache entry that is older than some threshold time τ in seconds is removed. This threshold determines the maximum temporal span of the moving average from equations (3.2) and (3.3) and the maximum response time for dynamic changes in the scene configuration. A larger value for τ increases the accuracy for the estimate of $I_i(\omega)$ by using a bigger averaging window and more rays. However, this may not be consistent with the current scene configuration if sources, listeners, or objects in the scene change position abruptly. On the other hand, a small value for τ requires more rays to be traced per frame to maintain accurate output, since the temporal averaging for values stored in the cache will have less effect. For $\tau = 0$, our technique gives the same result as traditional Monte Carlo path tracing.

Incremental Computation: The diffuse path cache incrementally computes a moving average of the incident intensity $I_i(\omega)$ for each sequence of surface patch reflections that arrive at the listener. We sample these values using traditional path tracing, but use a radiosity-like subdivision to take advantage of the coherence of rays from previous frames and to group the rays based on the sequence of reflections that have occurred. By grouping rays over many frames and reusing those results, we reduce the noise in the Monte Carlo estimation, yet need far fewer rays emitted from the sound sources, thereby reducing the time needed to compute diffuse reflections for interactive applications. Like radiosity-based algorithms, our method converges to traditional diffuse path tracing with a suitably small subdivision resolution λ. However, if λ is too small, it may require a greater number of rays to be traced and a larger diffuse path cache. In this case, fewer rays are grouped together and the effect of path reuse is reduced, resulting in a smaller benefit over traditional path tracing.

3.2.3 Impulse Response Cache

In the previous section, we presented a method that utilizes temporal coherence for diffuse reflection paths using the diffuse path cache (DPC). While the DPC is efficient for low-order diffuse reflections and provides detailed information about each propagation path, it is less suitable for high-order ray tracing where the number of possible reflection paths, and therefore the cache, becomes very large. In the case of late

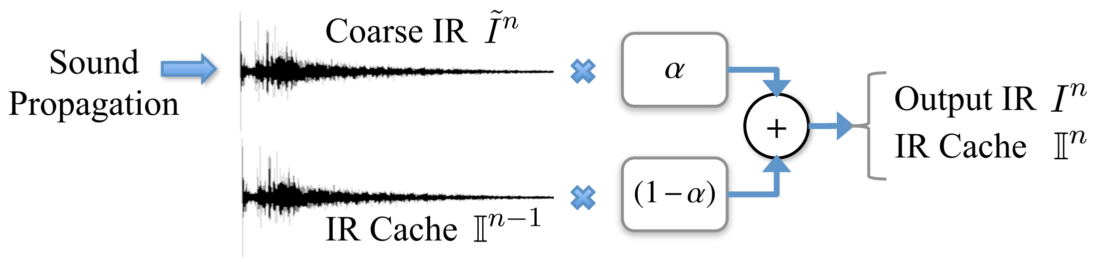


Figure 3.4: This figure summarizes our IR cache algorithm. A persistent copy of the IR, $\mathbb{I}^{n-1}$, is stored for each sound source. A small number of rays are used to compute a coarse impulse response for frame n, $\tilde{I}^n$, and this coarse IR is combined with the IR cache $\mathbb{I}^{n-1}$ from frame $n-1$ using the exponential smoothing factor α. The resulting filtered IR is then stored in the IR cache for the next frame before it is used for sound rendering.

reverberation with 50-200 orders of reflection, the number of paths that need to be stored can number in the millions per sound source. This results in both a significant performance and memory overhead that limits the interactivity of the diffuse path cache on complex scenes.

To address these issues, we introduce the notion of the *impulse response cache*, $\mathbb{I}_{\omega}^{n-1}(t)$, a copy of the last impulse response that is used to filter the output of a path-tracing based sound propagation algorithm. The IR cache $\mathbb{I}_{\omega}^{n-1}(t)$ stores the accumulated weighted sum of past impulse responses, and so uses multiple frames of path tracing to compute the resulting final IR for frame n, I^n. Our approach is general and can be applied to any sound propagation algorithm that uses a stochastic method such as Monte Carlo path tracing to compute an impulse response. During each frame, a different set of uniform random rays is traced, producing a slightly different impulse response. The quality of the impulse response and the computation time is dependent on the number of rays. The weighted sum of many of these impulse responses is a better estimate of the actual sound field than the IR computed only for that frame alone, since it contains the contributions of many more sound paths.

Our IR cache module takes as input an impulse response from path tracing, $\tilde{I}_{\omega}^{n}(t)$, that contains the contributions from a small number of rays traced on the current frame. The module produces a filtered impulse response utilizing the history information stored in the IR cache $\mathbb{I}_{\omega}^{n-1}(t)$. This process is summarized in Figure 3.4. We also introduce a parameter $\alpha \in [0,1]$ that controls how responsive the IR cache is to changes in the impulse responses. The final impulse response on frame n is computed by the recursive relationship in equation 3.5.

$$I_{\omega}^{n}(t) = \mathbb{I}_{\omega}^{n}(t) = \alpha\tilde{I}_{\omega}^{n}(t) + (1-\alpha)\mathbb{I}_{\omega}^{n-1}(t). \tag{3.5}$$

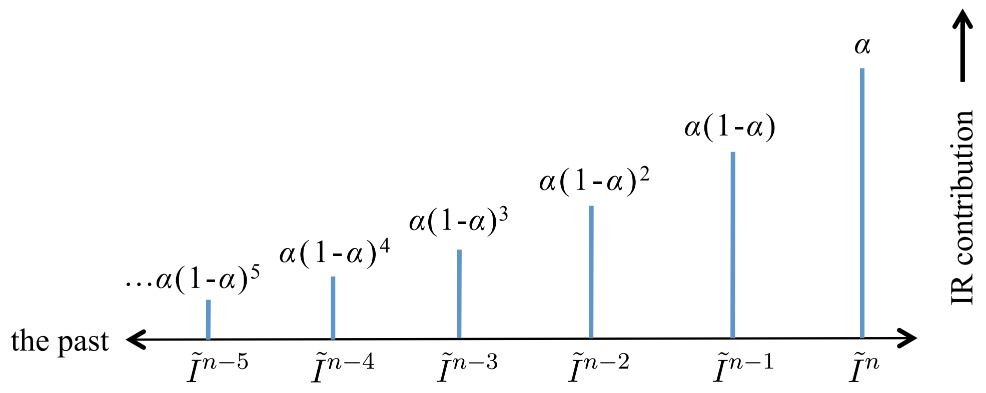


Figure 3.5: The contribution of the impulse responses from previous frames decreases for frames that are further in the past. The coarse impulse response from the current frame, $\tilde{I}^n$, contributes with weight α, the last frame $\tilde{I}^{n-1}$ contributes with weight $\alpha(1-\alpha)$, and the jth previous frame with weight $\alpha(1-\alpha)^j$.

The final IR $I^n_\omega(t)$ is a linear combination of the current frame's path tracing output, $\tilde{I}^n_\omega(t)$, and the contents of the IR cache for the sample, $\mathbb{I}^{n-1}_\omega(t)$. This is an application of the well-known temporal coherence technique called *exponential smoothing* (Brown, 1956). In graphics, exponential smoothing has previously been used to reduce the overhead of shading (Nehab et al., 2007), as well as to reduce spatial and temporal aliasing in shadow maps (Scherzer et al., 2007). In essence, the IR cache applies a 1st-order recursive low-pass filter to each sample in the impulse response, thereby using the history stored in $\mathbb{I}^{n-1}_\omega(t)$ to produce a higher-quality impulse response with less Monte Carlo noise artifacts and smoother variation over time. The parameter α determines the weight of the current IR $\tilde{I}^n_\omega(t)$ in the final output IR. A value of α close to 1 means that the system is more responsive to dynamic changes in the scene, but with less benefit from the IR cache. A value of α closer to 0 indicates that more weight is given to the IR cache, and thus the simulation will benefit more from the cache but be less responsive to dynamic changes in the scene. The contribution of past IRs to the current frame is illustrated by Figure 3.5.

Since it is unintuitive to determine the value of α directly, we propose the following method to compute α for a filtering window τ. This filtering window, given in seconds, is chosen to correspond to how long a coarse IR $\tilde{I}^n_\omega(t)$ from a previous frame is considered to contribute to the final IR $I^n_\omega(t)$. From the recurrence relation in equation 3.5, the contribution of coarse IR $\tilde{I}^{n-j}_\omega(t)$ during the jth previous frame is $\alpha(1-\alpha)^j$. To compute a value of α, given τ and time step Δt, our goal is to compute α such that $\alpha(1-\alpha)^j \leq \epsilon$, where $j = \frac{\tau}{\Delta t}$. $\epsilon \in [0,1]$ corresponds to the weight where an IR is not considered contributing. In our simulations, we use $\epsilon = 0.0001$. There is no closed-form expression for α, but standard root-finding techniques can be applied to find real $\alpha \in [0,1]$. Alternatively, the contribution of an IR can be approximated as $(1-\alpha)^j$ by dropping the initial α factor. This produces the following analytical solution in equation 3.6.

$$\alpha = 1 - \epsilon^{\Delta t/\tau}. \tag{3.6}$$

This expression enables an approximate value of α to be efficiently computed for a given time step and τ.

Variable Response Time: While a constant value of τ for every impulse response sample may give satisfactory results, we propose an extension that allows τ to vary for different parts of the IR. Previous work has shown that early reflections are perceptually important to a listener, especially in dynamic scenes. Therefore, they should be updated more often than later parts of the IR that are less useful for localization and that change more slowly (Begault et al., 2001; Müller-Tomfelde, 2001). Late reverberation can also take advantage of more temporal coherence by using a longer response time. As a result, we choose a smaller value of τ towards the beginning of the IR and a larger value towards the end. One possible approach is to define τ to be proportional to the delay time d. In our implementation, we compute τ using the relationship in equation 3.7.

$$\tau(d) = max(\beta d, \tau_{min}). \tag{3.7}$$

The scale factor β determines how quickly τ increases relative to the delay time. We use $\beta = 2$. The value of τ is clamped to always be greater than some minimum τ_{min}. From the value of $\tau(d)$ at each IR sample, the value of α is computed with equation 3.6, then used to update the IR cache with equation 3.5. This formulation enables the impulse response filtering to be applied in a perceptually relevant manner with a faster response time for the earlier reflections. It is also possible to design more complex schemes that can allow for τ to vary in arbitrary ways.

Frequency Bands: Another important aspect of sound propagation includes the generation of frequency-dependent sound effects. Like many prior geometric sound propagation systems, our approach computes the impulse response in discrete frequency bands. The output of ray tracing is a time-domain histogram of the sound energy for each band, $\tilde{I}^n_{\omega}(t)$. Our technique operates directly on this representation and applies equation 3.5 to each band independently. When the convolution subsystem is updated, a pressure impulse response suitable for audio rendering is computed by combining the information in all frequency bands (Kuttruff, 1993).

Directional Sound: Directional and spatial sound effects are also important for sound localization and immersion (Barron, 1974). The IR cache can be extended to incorporate directional information for each sample using a *directional IR*. Our implementation uses one 3D Cartesian vector per IR sample and denotes the directional IR computed on the current frame using the symbol $\tilde{\vec{x}}^n(t)$. The length of a vector in the direction IR indicates the strength of the directivity for that sample, and the direction of the vector indicates the average direction of sound arrival in world space. Each ray that is accumulated in the IR during sound propagation adds its unit-length direction vector to the directional IR and is weighted by the ray's intensity. The IR cache is augmented with a cache of the previous directional IR, $\vec{\mathbb{X}}^{n-1}(t)$. To update the directional part of the IR cache, we apply equation 3.5 to the x, y, and z components of each vector independently. Because the directional information in the IR will vary at a slow rate controlled by τ, we store all directional information in the world coordinate frame. For sound rendering, we transform the direction for each IR sample into the listener's local coordinate space for sound spatialization. This allows the listener's spatial sound to be updated at a rate faster than τ.

3.2.4 Impulse Response Length

A prominent feature of many geometric sound propagation algorithms is the need to choose a maximum order to which sound reflections are computed. The number of bounces required to capture a complete impulse response often varies widely for scenes of different sizes, shapes, and material properties. In addition, interactive simulations must consider the effects of dynamic objects in the scene that may alter the reverberation time, such as the opening or closing of a door. Outdoor scenes and coupled rooms can also be challenging for reverb time estimation (Carvalho, 1995). Many existing systems either specify a maximum distance or time that sound rays are propagated, or choose an arbitrarily high maximum reflection depth at which to truncate the response (e.g. 200). Other systems avoid computing certain inaudible paths based on time and spatial incidence (Begault, 1996), but will not work for diffuse late reverberation, which corresponds to the sum of many individually inaudible paths. Due to the dependence of the reverberation time on a variety of factors, it can be hard to predict the IR length that should be computed for general scenes. To make matters worse, such a system can result in poor performance by tracing rays to higher reflection order than may be audible to a human listener.

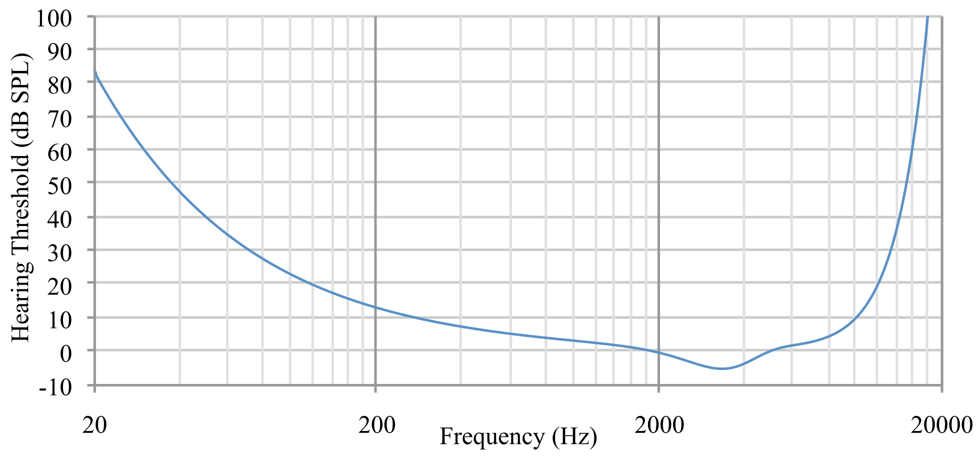


Figure 3.6: **Psychoacoustic Metric**: The human threshold of hearing varies greatly with frequency and is well-approximated by equation 3.8. The highest sensitivity is in the 2kHz $-$ 3kHz band, and the sensitivity decreases for very low and very high frequencies. Note that the horizontal axis is on a logarithmic scale.

We propose a novel approach that uses a psychoacoustic metric to dynamically optimize the ray propagation depth for interactive applications. We use information about the impulse response length from the previous frame $(n-1)$ to determine how far to propagate rays on the current frame (n). This feedback mechanism automatically adapts to changes in the response length, and thereby avoids parameter tuning. In addition, our approach computes the IR length for each source independently based on its sound power and the human threshold of hearing. This enables significant savings in computation for sources that are distant from the listener or quiet. For very loud sources, our approach automatically adjusts the IR length to capture the complete extended reverb decay. Therefore, our adaptive IR length algorithm can automatically handle a wide range of scenes and sound sources without any manual tuning of the IR length. In order to determine the audible length of a given impulse response, we make use of the *absolute threshold of hearing*. The human threshold of hearing corresponds to the smallest sound pressure level that can be perceived by a human listener and is a well-studied topic in psychoacoustics literature (Fletcher, 1940; Robinson and Dadson, 1957). The threshold, T_q, can be well-approximated over the audible frequency range for the average adult listener by equation 3.8 (Terhardt, 1979).

$$\begin{aligned} T_q(\omega) =& 3.64(\omega/1000)^{-0.8} - 6.5e^{-0.6(\omega/1000-3.3)^2)} + \\ & 10^{-3}(\omega/1000)^4. \text{ (db SPL)} . \end{aligned} \tag{3.8}$$

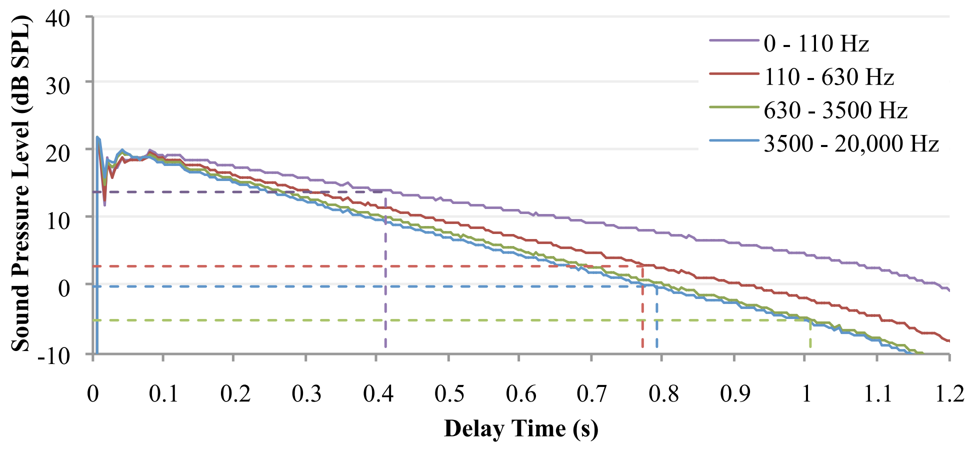


Figure 3.7: An example impulse response that shows the audible IR length for 4 frequency bands. Horizontal dashed lines correspond to the threshold of hearing for each band, and vertical dashed lines correspond to the IR length per-band. In this case, the maximum IR length over all bands is 1.01s. On the next time step, rays will be propagated for up to $1.01\text{s} + \Delta t_p$, where Δt_p is a parameter that determines how much the IR length can change per-frame.

This equation is used to compute the threshold in sound pressure level (dB SPL) for a frequency ω in Hertz. A visualization of the threshold in Figure 3.6 shows that the threshold varies greatly over the audible frequency range. We use this formulation to compute the threshold at runtime for a given simulation frequency band, and use the minimum value across each band as the threshold when determining the audible length of an IR.

As an input, our approach takes the full-length impulse response for each source consisting of an IR for each frequency band. Starting at the end of the IR for a given frequency band, we iterate over the IR samples in reverse and determine the last sample in the IR that is over the threshold of hearing. The delay time of this sample is reported as the perceptual IR length in seconds, t_p, for the frequency band. The results of this perceptual thresholding for an example IR are shown in Figure 3.7. In effect, there is no need to trace rays that are further than t_p time length, since those rays paths will be inaudible. To accomplish this, our algorithm stores this IR length information for each source so that it can be used to determine how far (in terms of propagation delay time) rays should be traced on the next frame. However, it is not sufficient to simply trace rays up to t_p, since this can cause the IR to artificially shrink in length over time, and it does not allow the IR to grow in length if the listener enters a more reverberant acoustic space. In order to address this problem, we always trace rays slightly past t_p. As rays are propagated through the scene, they are allowed to travel up to $t_{max} = t_p + \Delta t_p$ seconds, where Δt_p is a parameter that limits how much the IR length can grow or shrink during a given frame.

The larger the value of Δt_p, the quicker the algorithm can react to a change in the IR length. However, this is at the expense of tracing rays further past t_p. In our simulations, we use $\Delta t_p = 2\Delta t$, for simulation time step Δt. Therefore, if $\Delta t = 100$ms, we always trace rays for $t_{max} = t_p + 200$ms. This enables the IR length to grow or shrink by two seconds for every one second of real time, but still keeps extra ray propagation budget to a reasonable amount.

Once the value of t_{max} is computed, this information is stored and used to control the rays during the next frame for each source. When rays are propagated, no paths are computed that have a length greater than t_{max}. This results in significant savings for sound sources that are distant or quiet, and also allocates additional ray tracing for loud sources that require longer impulse responses. By computing an adaptive IR length, our approach also optimizes the ray tracing budget for scenes with different reverberation times.

3.3 Implementation

Sound Propagation We compute sound propagation for a multiple-of-four discrete frequency bands (4, 8, 12, etc.). This enables efficient use of SIMD vector CPU instructions. In our implementation, we use these logarithmically-distributed frequency bands: $0 - 110$Hz, $110 - 630$Hz, $630 - 3500$Hz, and $3500 - 22050$Hz. Our system uses different ray tracing approaches for computing specular early reflections and diffuse late reverberation. For early reflections, we use the ray-based image source method to find specular propagation paths up to a low reflection order (e.g. 5 bounces). The specular path cache ensures that the resulting paths are temporally coherent. For the late reverberation, we use Monte Carlo path tracing from the listener (Schissler and Manocha, 2016b) and augment it with the diffuse path cache and the impulse response cache to reduce the noise in the Monte Carlo estimation. Rays are propagated until the end of the impulse response is reached, as determined by our adaptive IR length approach. We make use of *diffuse rain* sampling (Schröder, 2011) to increase the number of sound paths found by generating an additional reflection to the sound source from each ray intersection point. The number of primary rays traced on each frame from the listener is calculated based on the time taken to compute the previous frame. This allows our system to adaptively reduce or increase the simulation quality to maintain a specific update time for sound propagation. About $500 - 1,000$ primary rays are traced for indoor scenes, while more rays are traced outdoors because most rays escape the scene after a few bounces. The sound propagation impulse response is spatialized using either vector-based amplitude panning (Pulkki et al., 2001) or head-related transfer functions.

The resulting spatial IR is convolved with the anechoic audio for each sound source to generate the final sound.

Specular Path Cache We implemented the specular path cache using a simple hash table with linked-list buckets. The hash code that uniquely identifies a path in the cache is incrementally computed as each ray propagates through the scene. The hash for a path of length d between source S_k and listener L_l is given by the expression $kl \sum_{i=0}^{d}(i+1)T_i$, where T_i is the hash code for the ith triangle that was hit along the ray path.

Diffuse Path Cache The diffuse path cache is also implemented using a hash table, and the hash code for a path is computed in a similar way as for the specular path cache. For the subdivision used to group diffuse paths, we used a subdivision size of $\lambda = 0.5m$ for our simulations. We give more details on the accuracy of the algorithm as a function of patch size (λ) in Section 3.4.2.

Impulse Response Cache The IR cache is implemented as a simple large array of impulse response samples. Each sample consists of an intensity frequency response (e.g. intensity at 4 frequency bands), plus the average directivity for that sample (e.g. Cartesian direction or low-order spherical harmonics). The data for each sample are stored contiguously in memory. The update to the IR cache (Equation 3.5) can be efficiently implemented using SIMD instructions and a few multiply-add operations per sample.

3.4 Results

In this section we present results for our temporal coherence sound propagation techniques. First, we describe the results for our caching approach for specular reflections using the ray-based image source method. Then, we show how caching diffuse paths can be used to improve the quality of Monte Carlo path tracing. Lastly, we present results for the impulse response cache and adaptive IR length approaches.

3.4.1 Specular Path Cache

The specular path caching approach was evaluated on four different scenes, depicted in Figure 3.8. The scenes all have one sound source, a moving listener, and have geometric complexity typical of interactive VR and games. The main results on these scenes are shown in Table 3.1. The performance results were measured using a single thread running on an Intel i7 4770k CPU. We compared our approach that uses a specular path

	Complexity		**RISM, 10k rays**		**RISM, 1k rays**		**RISM+SPC, 1k rays**	
Scene	#Tris	#Src.	Avg. #Paths	Time (ms)	Avg. #Paths	Time (ms)	Avg. #Paths	Time (ms)
Auditorium	122,539	1	9.7	79.5	3.4	8.7	7.9	**8.9**
Indoor	1,528	1	44	31.5	13.8	3.2	47.8	**3.4**
Sibenik	57,892	1	16.3	64.7	6.7	7.8	16.8	**7.5**
Sponza	104,492	1	52.3	77.9	16.9	8.7	53.2	**8.8**

Table 3.1: The main results of our specular path cache approach on four benchmark scenes with 10th-order specular reflections. We compare the performance of the naïve ray-based image source method (RISM) with 10,000 and 1,000 primary rays to our approach with 1,000 rays (RISM+SPC). Our approach generates about as many propagation paths as RISM with 10,000 rays, but has performance close to the RISM with 1,000 rays.

cache (known as RISM+SPC) to the naïve ray-based image source method (RISM) with different numbers of primary rays. Previous systems that utilize the RISM algorithm typically use 10,000 - 50,000 primary rays (Vorländer, 1989; Taylor et al., 2012). We compare the RISM with either 10,000 rays or 1,000 rays to our approach with 1,000 rays.

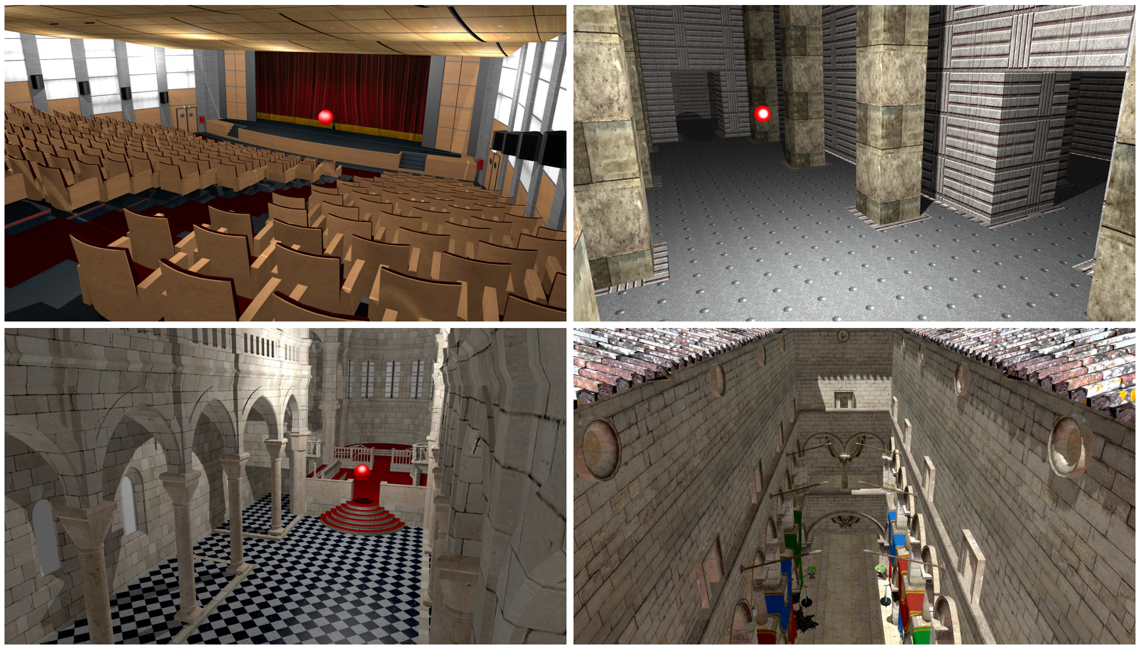


Figure 3.8: The four benchmark scenes used to evaluate the specular path cache approach: Auditorium (top left), Indoor (top right), Sibenik (bottom left), Sponza (bottom right).

In terms of the number of specular reflection paths that are detected, our approach finds about as many paths as the RISM with 10,000 rays. On all scenes except for the auditorium, our approach with 1,000 rays generates more paths than the naïve RISM with 10,000 rays, yet uses about 11% of the computational resources. In comparison to the RISM with just 1,000 rays, our method with 1,000 rays is able to detect

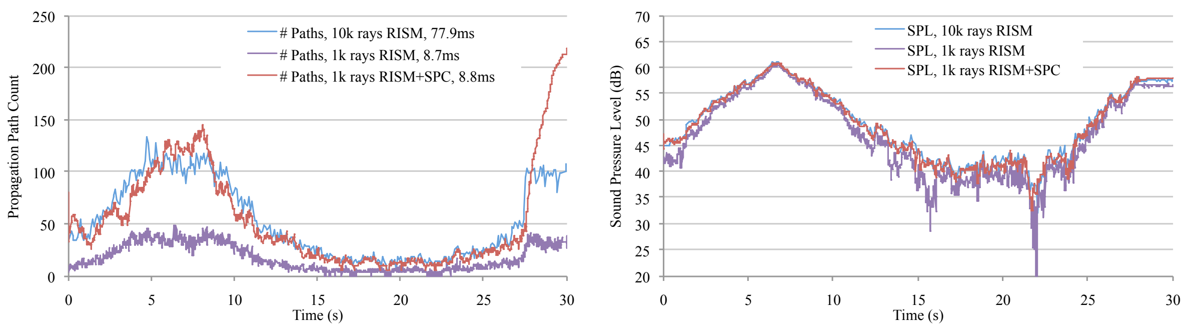


Figure 3.9: Our specular path cache (SPC) algorithm significantly improves the quality of the ray-based image source method (RISM) in terms of the number of propagation paths generated (left) and the sound pressure at the listener's position (right) in the Sponza scene. With only 1,000 primary rays, our approach produces nearly the same output as the naïve RISM with 10,000 rays, but with about the same cost as the naïve RISM with 1,000 rays.

about $2-4$x as many specular reflection paths, even though the computational cost is nearly the same. The variation in the number of paths over time is illustrated for the Sponza scene in Figure 3.9. For the majority of the scene, our approach with 1,000 rays finds about the same number of paths at the RISM with 10,000 rays, whereas the RISM with 1,000 rays finds about 3x fewer paths. The improvement provided by our method depends significantly on the amount of dynamic motion in the scene. When there is motion (e.g. $t = 0$ to $t = 27$), our approach reuses some propagation paths from previous frames, but many of the paths from previous frames must be discarded to avoid introducing any error. However, when the scene is mostly static (e.g. $t = 27$ to $t = 30$), our approach finds significantly more paths than the RISM with 10,000 rays, and continues to add new paths to the cache until they become invalid. As a result, in static scenes our approach can continue to improve the sound quality on each frame for as long as it is executing.

The use of temporal coherence improves the subjective sound quality, since variation in the sound field from frame to frame is reduced. The naïve RISM generates a different set of rays on each frame, and therefore can detect a different set of paths, leading to unnatural variation in the sound, especially when the source and listener are static. These differences in subjective sound quality are most noticeable when spatial rather than monaural sound is rendered. The variation in sound pressure for the three methods is depicted in Figure 3.9. With just 1,000 rays, the naïve RISM has significant variation in the sound level from $t = 10$ to $t = 25$. At $t = 22$, the sound cuts out completely. However, both RISM with 10,000 rays and our method with 1,000 rays have fewer discontinuities. This variation produces audible artifacts in the rendered sound, and is avoided by our method at an insignificant additional cost.

In terms of memory, our approach only requires a small amount of storage for the specular path cache. A 10th-order reflection path requires only a few hundred bytes to store the path identifier K, and the number of paths in the cache for each source is generally less than 1,000. As a result, our technique uses at most a few hundred kilobytes per sound source, and usually significantly less than that amount.

These exceptional results are due to the use of the specular path cache (SPC). The SPC stores sound propagation paths that were found to be valid on previous simulation updates, and then re-validates the cached paths on each time step. Rather than recomputing the entire set of propagation paths on each frame, our approach iteratively refines the propagation paths based on the current scene configuration and new rays that are traced. Since the cost of validating a specular reflection path is orders of magnitude less than the cost involved in detecting a path (Vorländer, 1989), our method produces a large savings in computation time. Our approach using the specular path cache is able to compute 10th-order specular reflections in dynamic scenes in just $3.4 - 8.9$ms, whereas the previous method uses $31.5 - 79.5$ms to generate sound with similar quality.

3.4.2 Diffuse Path Cache

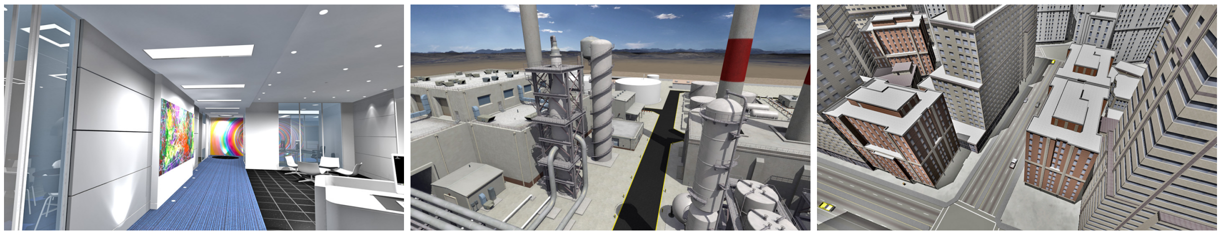


Figure 3.10: The three benchmark scenes used to evaluate our diffuse path caching algorithm. (left) office interior (154K triangles); (center) oil refinery (245K triangles); (right) city (254K triangles).

We evaluated the diffuse path cache approach on large indoor and outdoor scenes with high complexity. The scenes are shown in Figure 3.10. Our approach can generate plausible diffuse reflection effects at interactive rates on a 4-core Intel i7 4770k CPU (see Table 3.2).

Office: The listener walks through a moderate-sized office environment with six sound sources. Diffuse reflections area significant component of this indoor scene due to material properties with high scattering coefficients.

Refinery: This large outdoor scene of an oil refinery demonstrates the ability of our diffuse reflection approach to scale to large outdoor environments.

Scene	Scene Complexity #Tris	#Sources	#Specular	#Diffuse	Performance Total Time (ms)
Office	154,020	6	10	10	33.6
Refinery	245,828	4	10	10	27.4
Small City	89,792	8	10	6	19.3
Large City	254,903	14	10	6	71.0

Table 3.2: The main performance results of our diffuse path cache approach. We show the complexity of the various benchmark scenes, the number of sound sources, and the number of specular and diffuse reflection bounces. The diffuse path cache can be used to compute sound propagation for diffuse reflections at interactive rates.

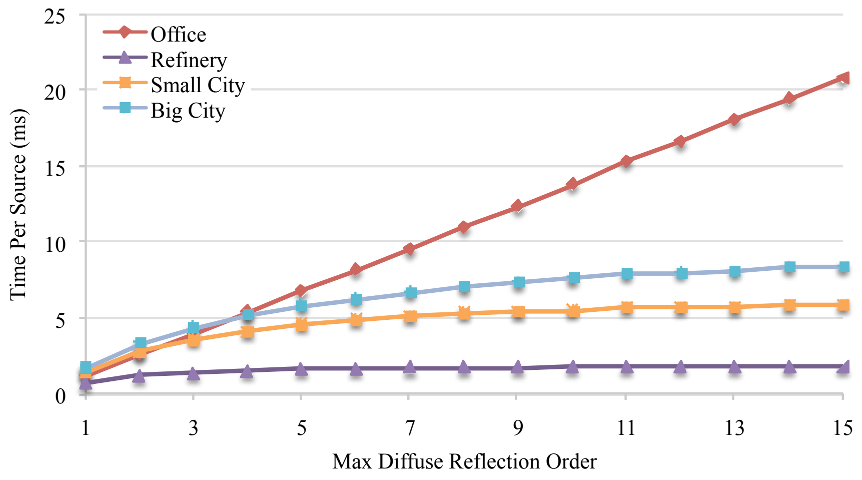
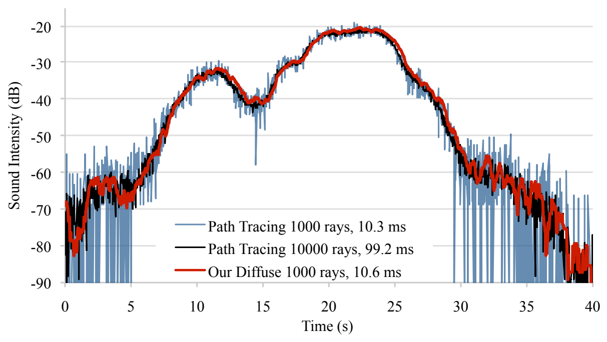


Figure 3.11: We highlight how the performance of our diffuse path cache algorithm scales with the maximum diffuse reflection order on a single CPU thread. Note that in outdoor scenes, most rays escape the scene after the 4th or 5th bounce. In the indoor office scene, the complexity is linear with respect to the maximum diffuse order.

Small City: This small city scene demonstrates that our approach can handle both moving sources, listeners, and obstacles. The listener sits in a vehicle that drives through a city scene, where it passes other cars that are moving sound sources as well as obstacles.

Big City: A pedestrian listener walks along the sidewalk at an intersection in a large city model with more than 50 buildings over an area of 0.5 km^2. There are 14 moving sound sources (cars, ambulance) that pass by the pedestrian listener. These vehicles are also moving obstacles. This scene shows how our approach can scale well with the number of sources and maintain interactive frame rates for large scenes.

We have analyzed the runtime performance as well as accuracy of our diffuse reflection computation algorithms. Fig. 3.11 shows that the performance of our algorithm scales linearly for increasing maximum diffuse reflection order. In practice, our incremental algorithm is able to simulate over 10 diffuse reflection bounces in the benchmarks at around $50-60$Hz for a single sound source. We also compared the accuracy



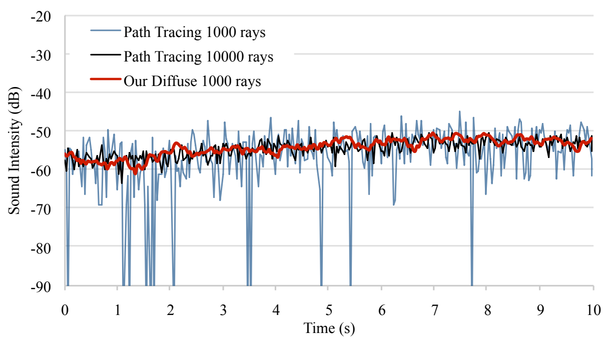


Figure 3.12: We compare the sound intensity received at the listener computed by our diffuse path cache method with that of the traditional path tracing algorithm in the office benchmark. The accuracy of our algorithm (with 1000 rays) is comparable to the path tracing algorithm (using 10000 rays) because it utilizes temporal coherence. On the other hand, path tracing with 1000 rays misses many paths and results in inaccurate diffuse sound with significant noise and variation over time.

of our algorithm with traditional path tracing (see Fig. 3.12). While we use 1000 rays with our approach, we compare its accuracy with two types of path tracing: 1000 rays and 10,000 rays, and compute 10 orders of reflection. The accuracy of our algorithm is comparable to that of path tracing with 10,000 rays, with an average error of 2.27 dB. On the other hand, path tracing with only 1000 rays, results in noisier results an average error of 6.69 dB. The amount of noise is strongly influenced by the geometry of the scene and how occluded the source and listener are from each other. The temporal averaging of our method dramatically improves the results for a given number of emitted rays (i.e. 1000 rays). Our approach is effective at improving the accuracy of low-intensity sound in the left and right portions of the graph. We average the contributions of many previous frames to compute a better estimate of the sound intensity than is possible with a naïve approach and the same number of rays.

Response Time: In Figure 3.13, we examine the effect on convergence of varying the response time parameter, τ, in the office scene with 1000 rays per frame. We measure the error in our approach for different values of τ when compared to the ground truth of brute-force path tracing with 20k rays. With $\tau = 0$, we observe similar results to naive path tracing with 1000 rays because no diffuse path caching occurs. As τ increases, the accuracy of our approach increases due to a larger averaging window that uses the ray contributions from more frames. However, our approach may introduce some error for large values of τ in dynamic scenes where there are abrupt changes in the sound intensity received at the listener. Figure 3.13 shows the time-domain smearing of the sound energy due to the averaging effect. When τ is large, the resulting audio is very smooth

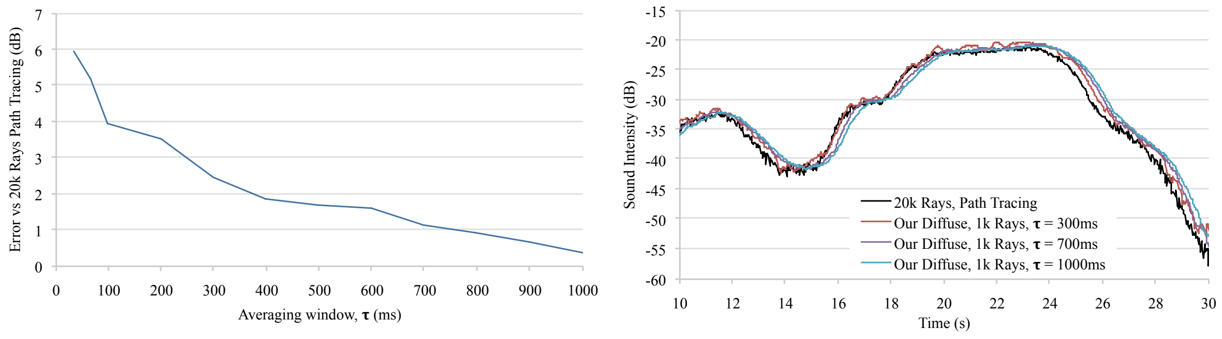


Figure 3.13: The choice of parameter τ determines how quickly the diffuse path cache will react to a change in the scene configuration. By using a longer averaging window, τ, we demonstrate that our diffuse approach converges to a small error when compared to brute-force path tracing (left). For increasing τ, we notice that there is a reaction delay of approximately τ versus naïve path tracing when the scene is dynamic (right). Changing τ has insignificant impact on the runtime performance of our approach.

and free of sampling noise, but at the expense of slower reaction to quick changes in the scene. We found experimentally that $\tau = 300$ms produced the best balance between utilizing ray coherence and time-domain accuracy. It may also be necessary to clear the cache if there is a large sudden change in the scene in order to reduce the error.

Subdivision Resolution: We also examined the effect of varying the size of λ, the surface patch subdivision resolution. In Figure 3.14, we observe no change in the response time or accuracy due to varying values of λ when compared to standard path tracing. However, very large values of λ (e.g. ≥ 1 m) may result in significant error in the delay time or the listener-relative direction for propagation paths. Because bigger subdivisions group together more rays that are incoherent and may have different path lengths, the time-domain accuracy or directionality of propagation paths may be affected. These errors are generally imperceptible for diffuse sound because individual propagation paths cannot be distinguished.

Comparison with Previous Works: Most prior geometric techniques for diffuse reflections are based on Monte Carlo path tracing (Embrechts, 2000; Lentz et al., 2007). We have compared the accuracy as well as runtime performance with path tracing algorithms in Figure 3.12. The main benefit of our method arises from the fact that we can shoot almost one order of magnitude fewer rays as compared to path tracing and achieve similar accuracy. This is due to the fact that we perform temporal averaging that can significantly improve the accuracy. The RESound system (Taylor et al., 2009) takes about $250 - 500$ms to compute up to

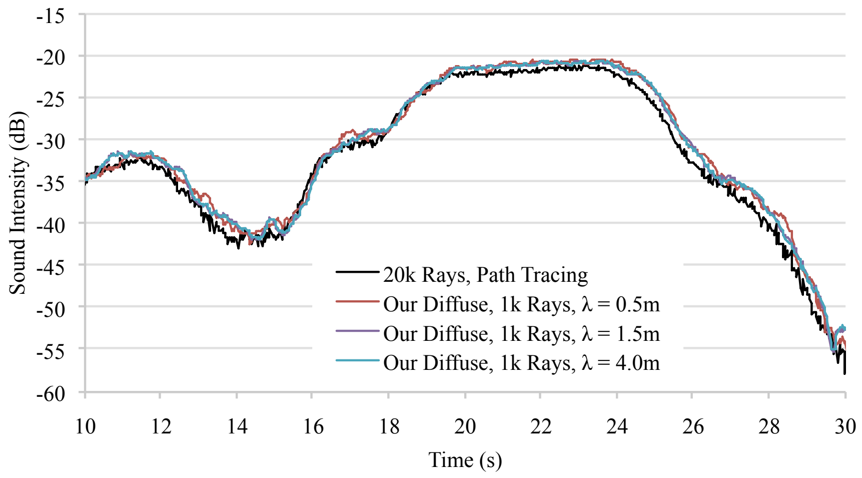


Figure 3.14: This graph shows the sound intensity received at the listener for different values of λ, the diffuse path cache subdivision size. Even large values of λ produce no significant changes in the accuracy or response time of our algorithm.

3 orders of diffuse reflections (with 200K rays) on models with $60 - 280$K triangles using seven threads on a multi-core CPU. On the other, our algorithm takes less than 15ms per source to compute up to 10 orders of diffuse reflections.

3.4.3 IR Cache & Adaptive IR Length

We evaluated the IR cache and adaptive IR length approaches on complex dynamic indoor and outdoor scenes typical of those found in games and interactive applications (Figure 3.15). We achieve real-time performance for dozens of sound sources on a 4-core Intel i7 4770k desktop CPU. These results are summarized in Table 3.3.

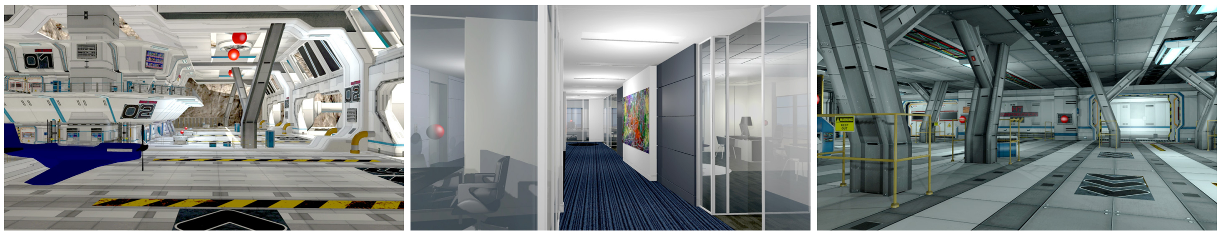


Figure 3.15: The three benchmark scenes used to evaluate our ir cache and adative IR length algorithms. The scenes have dozens of sound sources and dynamic elements (e.g. doors) that alter the sound propagation. In the hangar scene, we show a fast-moving aircraft sound source to demonstrate how the perceptual impact of our temporal coherence technique is small.

	Complexity		**Propagation**				**Time (ms)**		
Scene	#Tris	#Src.	#Rays	#Bounces	IR Length (s)	Mem. (MB)	Ray Tracing	Cache	Total
Space Station	35,581	21	588	136	0.8 - 1.25	40.8	105.0	3.42	**111.57**
Office	82,125	24	680	129	0.5 - 1.0	39.3	69.0	2.70	**74.08**
Hangar	71,461	18	1094	95	1.5 - 3.0	75.3	100.5	6.10	**110.91**

Table 3.3: We highlight the primary results of our IR cache algorithm. Our approach is able to achieve interactive performance on complex scenes with dozens of sources. The IR cache uses only a few MB of memory per sound source, while the adaptive impulse response length ensures that only audible rays are traced when the IR length changes.

Hangar: This aircraft hangar with 18 sound sources demonstrates the ability of our approach to dynamically determine the IR length for different acoustic spaces and source power levels. As the listener moves into the open outdoor hangar, the reverb time increases for loud aircraft sound sources. Due to these loud sources and different environments, the IR length changes from 1.5s to 3.0s.

Office: The listener walks through a large office environment with 24 sound sources that include human voices, moving doors, running water, and an elevator. The IR length varies from 0.5s to 1.0s.

Space Station: This scene is set on a space station in Earth orbit. There are 21 sound sources that include ventilation, computers, radios, and other ambient sounds. We show that our system can handle the effects of dynamic geometry (doors) on reverberation time. In this scene the IR length varies from 0.8s to 1.25s.

Performance: On these complex scenes, our interactive sound propagation system is able to compute high-quality sound in 74ms $- 111$ms. The use of the IR cache enables just $500 - 1000$ rays to be traced per frame. As a result, the time spent in ray tracing for our method is interactive for scenes with around 20 sources. The additional time required to update the cache using equation 3.5 is only a few milliseconds and scales linearly with the length of the IR.

We also compared the performance of our adaptive IR length approach relative to that for a static IR length. The static IR length was chosen by hand for each scene to be the longest audible IR length for the given scene. The lengths were 1.25s, 1.0s, and 3.0s for the Space Station, Office, and Hangar scenes respectively. Figure 3.16 shows that our adaptive approach provides a $30 - 60\%$ speedup over the static IR length. Since the audible IR length for a sound source (L_p) is often less than the maximum IR length for a scene, our method can reduce the computation required for quieter sources by tracing fewer ray reflections. The adaptive IR length can handle dynamically changing reverberation times and there is no longer the requirement to set the IR length parameter by hand. For any given situation, our approach computes only the

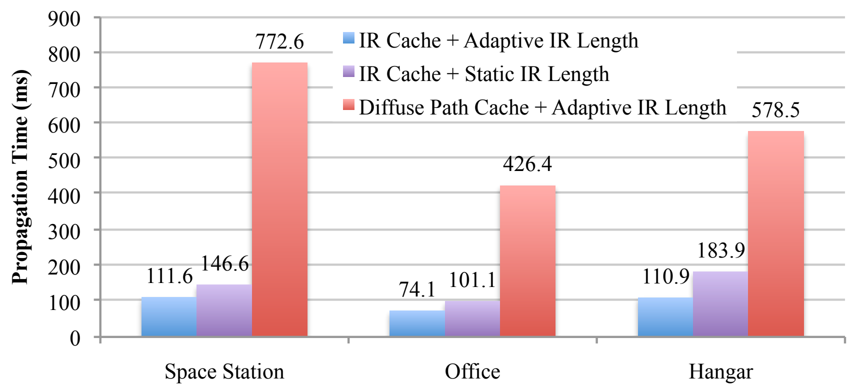


Figure 3.16: When compared to a static IR length chosen by hand for each scene, our adaptive IR length is $30 - 60\%$ faster for these benchmarks. The diffuse path cache of (Schissler et al., 2014) incurs significant overhead when compared to the IR cache approach. The ray tracing time is the same for all techniques.

impulse response samples that are audible to a human listener.

Comparison with Previous Works: In the supplementary video, we compare the sound generated by our method with 1500 rays to that of naive path tracing (Embrechts, 2000) with 90,000 rays. Using the IR cache, we achieve better sound quality using our method than path tracing with around 50x as many rays. Since path tracing computes the IRs for each frame independently without any history information, it is susceptible to Monte Carlo aliasing artifacts (e.g. unnatural variations in volume, spatial sound) unless a very large number of rays is traced. On the other hand, our approach produces IRs that change smoothly by taking into account many frames of history. At a cost of some responsiveness (controlled via τ) and extra storage for the IR cache, our system can generate high-quality sound using substantially fewer rays, thereby improving interactivity. In the video, we show how the sound quality changes as the value of τ varies from long (3.0s), to short (0.5s), and then finally 0.0s (equivalent to path tracing). The overall performance benefit of our method over standard path tracing is at least 50x.

Recent work in sound has used temporal coherence to improve the results of path tracing for interactive sound propagation. One technique, described in detail in Section 3.2.2, suggests a *diffuse path cache* that caches individual ray reflection paths and stores a moving average of the sound energy for each path in the cache (Schissler et al., 2014). This approach achieves roughly a 10x speedup over standard path tracing by using the diffuse path cache. However, the memory overhead required to store all of the paths (possibly millions) is significant. On these benchmarks, this method requires $200 - 300$MB of storage *per sound source*. Also, the diffuse path cache must eventually remove paths from the cache to keep from growing

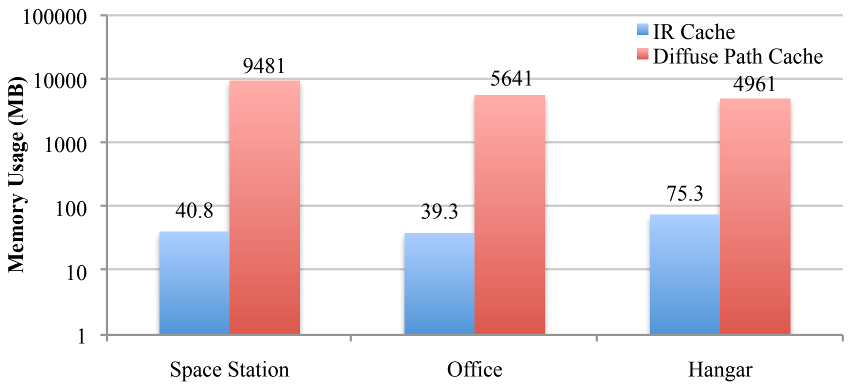


Figure 3.17: We compare the memory usage of our IR cache approach to the diffuse path cache of (Schissler et al., 2014) that stores individual ray paths. Our technique uses roughly 2 orders of magnitude less memory to compute sound of similar fidelity. Note that the vertical axis has a logarithmic scale.

too large. This process can alter the characteristics of the resulting sound, causing there to be slightly less reverberation than for a ground-truth simulation.

In comparison, the IR cache is roughly 5x faster than the diffuse path cache. We show the performance for each method on our benchmarks in Figure 3.16. The IR cache does not need to discard any past results, so effectively an infinite amount of history can be used with no additional computation or memory overhead by increasing the value of the response time τ. The memory required for the IR cache approach is only a few additional MB per sound source, and it scales linearly with the length of the IR. In Figure 3.17, we show that the IR cache uses around 2 orders of magnitude less memory than the diffuse path cache. In the video comparison, the IR cache achieves similar or better sound quality.

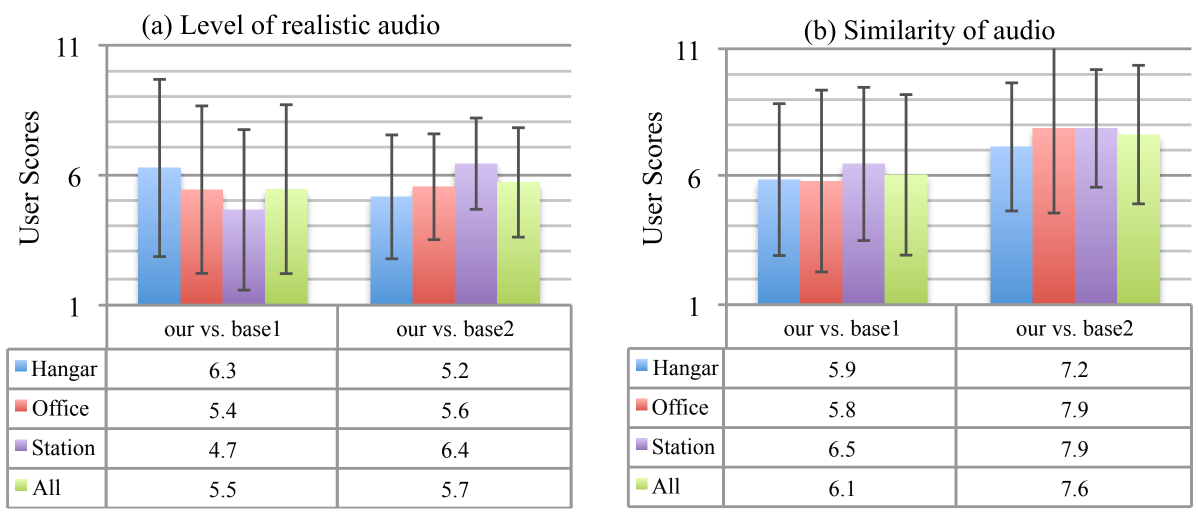
3.5 User Evaluation

We conducted a preliminary user evaluation to study the subjective impact of the IR cache and adaptive IR length approaches compared to previous methods. In the study, we compared the sound generated by our temporal coherence approach, called *our* method, to the sound generated when temporal coherence is disabled, called the *base* method. For the *base* method, we test two conditions. In the first, denoted by *base1*, the number of rays and performance is the same as for *our* method (see Table 3.3). In the second, denoted by *base2*, a ground-truth offline simulation is performed with 200,000 rays.

Study Design: In the study, we tested the impact of temporal coherence on 3 scenes: hangar, office, and space station. There are 2 comparison conditions used for each scene: *our* vs. *base1* and *our* vs. *base2*, for a total of 6 conditions. The study was conducted as an online survey where each subject was presented with a 6 pairs of identical videos with different audio in a random order and asked two questions about each pair. The questions were (a) "which video has the more realistic audio?" and (b) "how similar is the audio in the videos?". The responses were recorded on a scale from 1 to 11. For the first question, an answer of 1 indicates the left video was much more realistic, an answer of 11 indicates the right video was much more realistic, and an answer of 6 indicates the videos were equal. On the second question, an answer of 1 indicates the videos sounded extremely different, while an answer of 11 indicates the videos sounded very similar. Our research hypotheses were: (1) *Our* method produces sound that is as realistic as *base2*, and more realistic than *base1*; (2) The sound generated by *our* method will be more similar to that of *base2* than it is to *base1*.

Study Procedure: The study was completed by a total of 18 subjects between the ages of 20 and 31, made up of 15 males and 3 females. The average age of the subjects was 25.2, and all subjects had normal hearing. At the start of the study, subjects were given detailed instructions and filled out a demographic information questionnaire. Subjects were required to use either headphones or earbuds when taking the study, and they were asked to calibrate their listening volume using a test audio clip. Subjects were then presented the 6 pairs of videos and responded to the questions for each pair. The subjects were allowed to replay the videos as many times as needed, and they were able to move forward and backward in the study to change their answers if necessary. After rating the 6 video pairs, the study was completed.

Study Results: The results of our preliminary user evaluation are summarized in Figure 3.18. For question (a), the subjects indicated the technique that had the more realistic audio. For the comparison between *our* method and the *base1* method, our method is slightly preferred across all scenes except for the Hangar with average scores between 4.7 and 6.3. When compared to the offline ground-truth simulation of *base2*, the preference for *our* method is less, with scores between 5.2 and 6.4. A two-tailed one-sample t-test across all scenes was used to test hypothesis 1 that *our* method will be slightly more realistic than *base2* ($p = 0.23$), and much more realistic than *base1* ($p = 0.34$). Due to the statistical insignificance of these results we cannot rule out the possibility that the difference between the methods is due to other factors.



	our vs. base1	our vs. base2
Hangar	6.3	5.2
Office	5.4	5.6
Station	4.7	6.4
All	5.5	5.7

	our vs. base1	our vs. base2
Hangar	5.9	7.2
Office	5.8	7.9
Station	6.5	7.9
All	6.1	7.6

Figure 3.18: We compare the user evaluation results for *our* approach versus the *base1* and *base2* methods for 3 scenes and 2 comparison cases. For question (a), a score below 6 indicates a preference for the first method in the comparison, whereas a score above 6 indicates a preference for the second method. For question (b), the higher the score, the more similar the audio for the two methods under comparison.

For question (b), the audio similarity between the techniques was considered. The subjects judged *our* method to be most similar to the *base2* offline simulation with scores from 7.2 to 7.9, while *our* method was scored less similar to the *base1* interactive simulation with scores from 5.8 to 6.5. Hypothesis 2 was evaluated by comparing the *our* vs. *base1* and *our* vs. *base2* conditions with a two-tailed Welch's t-test. In this case, the results are more significant ($p = 0.11$), but do not conclusively support the hypothesis that our method is more similar to *base2* than it is to *base1*. Overall, the large standard deviations of the scores indicate that subjects have difficulty discerning the subtle differences between the methods.

3.6 Conclusion, Limitations, and Future Work

In this chapter we have presented an approach for interactive sound propagation that uses various forms of temporal coherence to improve both the quality and performance of ray tracing. We use a cache of specular reflection paths, the *specular path cache*, to reduce the cost involved in finding early reflections. For diffuse reflections, we use the *diffuse path cache* to filter the sound intensity for propagation paths over many frames, thereby reducing noise and the number of rays required. To efficiently handle high-order reflections, we proposed the *impulse response cache* as a means to utilize temporal coherence. Finally, we showed how the length of the impulse response can be adaptively estimated using a perceptually-based technique. We evaluated these algorithms on a variety of complex multi-source scenes that are typical of interactive applications, and observe a performance benefit of at least an order of magnitude versus previous ray-tracing approaches.

Limitations and Future Work: Our approach is based on ray tracing, and therefore all standard limitations of geometric acoustics are inherent to our formulation. These include potential inaccuracies for low frequencies where we assume that primitives are larger than the wavelength. Since our approach is fundamentally based on the assumption that the sound at the listener does not change very quickly, some error will be introduced when there is a fast dynamic change in the scene, such as when a source or listener moves quickly. Our approach takes a short amount of time to react in these situations, but this delay can be controlled by parameter τ. Under extreme changes in the scene (e.g. teleportation), the cache data structures can be reset to their initial state to avoid these artifacts. In the *diffuse path cache* approach, we use a surface subdivision to group propagation paths within the cache. If the size of the subdivision is too large it may cause error in the delay time or the direction of propagation paths. The *diffuse path cache* may accumulate too many entries if it is used for high-order reflections, since the number of possible propagation paths becomes large. This results in high memory usage and longer time needed to update the cache each frame. Due to differences in the hearing threshold of different individuals, the use of equation 3.8 may not necessarily apply to all users of our system, and so our adaptive IR length algorithm may not align well with every user. However, an accurate threshold can be determined for each individual through personalized measurements or evaluation.

One potential direction for future work would be to perform further subjective evaluation of the impacts of temporal coherence on sound propagation. In particular, detailed user studies with expert listeners are needed to discern the subtle differences in sound quality and to investigate the effect of dynamic scenes on the perceived latency. Another possible way that our temporal coherence approach could be improved is to use an adaptive response time τ. If the sound propagation module detects large changes in the impulse response, it could automatically reduce the response time to reduce smoothing artifacts. Correspondingly, it could increase the response time and make better use of temporal coherence if the impulse response is not changing. We would also like to investigate if sound source masking or other psychoacoustic effects can be used to enhance our sound propagation algorithm. For instance, quiet sound sources may be inaudible if there is a loud noise, and in this case some additional computation can be saved.

CHAPTER 4: ACOUSTIC MATERIAL CLASSIFICATION AND OPTIMIZATION [1]

4.1 Introduction

Recent advances in computer vision have made it possible to generate accurate 3D models of indoor and outdoor scenes from a sequence of images and videos. The resulting models are frequently used for visual rendering, physics-based simulation, or robot navigation. In many applications including computer-aided design, teleconferencing, and augmented reality, it is also important to augment such scenes with synthetic or realistic sound effects. It has been shown that good sound rendering leads to an improved sense of presence in virtual and augmented environments (Härmä et al., 2004; Larsson et al., 2010).

In order to simulate sound propagation within a real-world scene, a 3D surface representation is needed, usually in the form of a triangle mesh. Another important requirement for sound propagation is the need for accurate acoustic material properties for the 3D scene representation. These properties include absorption and scattering coefficients. They specify how sound waves interact with surfaces in the scene and can strongly influence the overall acoustic effects in the scene, including the reverberation time. The material properties depend on a variety of factors including the angle of sound incidence, frequency, acoustic impedance, thickness, surface roughness, and whether or not there is a resonant cavity behind the surface (Egan, 1988; Kuttruff, 2007; Seddeq, 2009). Current sound propagation techniques have relied on tables of measured acoustic material data that are assigned to the scene triangles or objects manually by a user (Rindel and Christensen, 2007). However, assigning these properties is a time-consuming process that requires an in-depth user knowledge of acoustic materials. Furthermore, the resulting simulations may not match known acoustic characteristics of real-world scenes due to inconsistencies between the measured data and actual scene materials.

Therefore, we introduce a novel technique for automatically determining the acoustic material properties of 3D-reconstructed real-world scenes for multimodal augmented reality applications. Our approach builds on recent advances in computer vision and 3D scene reconstruction and augments them with a few simple

[1]Much of this chapter appeared previously Schissler et al. (2017a)

acoustic impulse response measurements. We apply a convolutional neural network to the images of the real-world scene in order to classify the materials associated with each triangle of the 3D reconstructed model (Section 4.2.1). These materials are used to initialize an optimization algorithm that iteratively adjusts the frequency-dependent absorption coefficients until the resulting acoustic simulation, computed using path tracing, is similar to the measured impulse responses from the real scene (Section 4.2.2). The resulting 3D model and the acoustic material characteristics are used to simulate realistic sound propagation effects. This process is illustrated in Figure 4.1. We have evaluated our technique on several room-sized scenes and show that it is able to generate impulse responses that closely match the ground-truth measurements in those rooms (Section 4.4). We also present a preliminary user study that demonstrates the subjective plausibility of the sound produced by our algorithms (Section 4.5). To the best of our knowledge, this is the first approach for automatic computation of acoustic material properties from 3D reconstructed models for augmented reality applications.

4.2 Acoustic Material Classification and Optimization

In this section we describe our approach and how it enables sound propagation in 3D reconstructions of real-world scenes. An overview of the pipeline is shown in Figure 4.2. As input, our technique takes a dense 3D triangle mesh that has been reconstructed using traditional multi-camera computer vision approaches. We assume that the mesh is mostly free of holes and other reconstruction errors. Our pipeline begins by applying a CNN-based material classifier to each of the RGB camera images from the reconstruction to determine the probability that materials from a known database are present. The materials in each image are projected onto the 3D mesh and the most likely material is chosen for each material patch, where the patches are generated using a 3D superpixel segmentation algorithm. If acoustic measurements of the real scene (i.e. recorded audio samples) are available, this material information is used to initialize an optimization algorithm that iteratively refines the materials so that virtual sound propagation matches these acoustic measurements. The result is a 3D mesh and a set of materials that can be used for sound propagation and can generate virtual sounds that match those in the real environment.

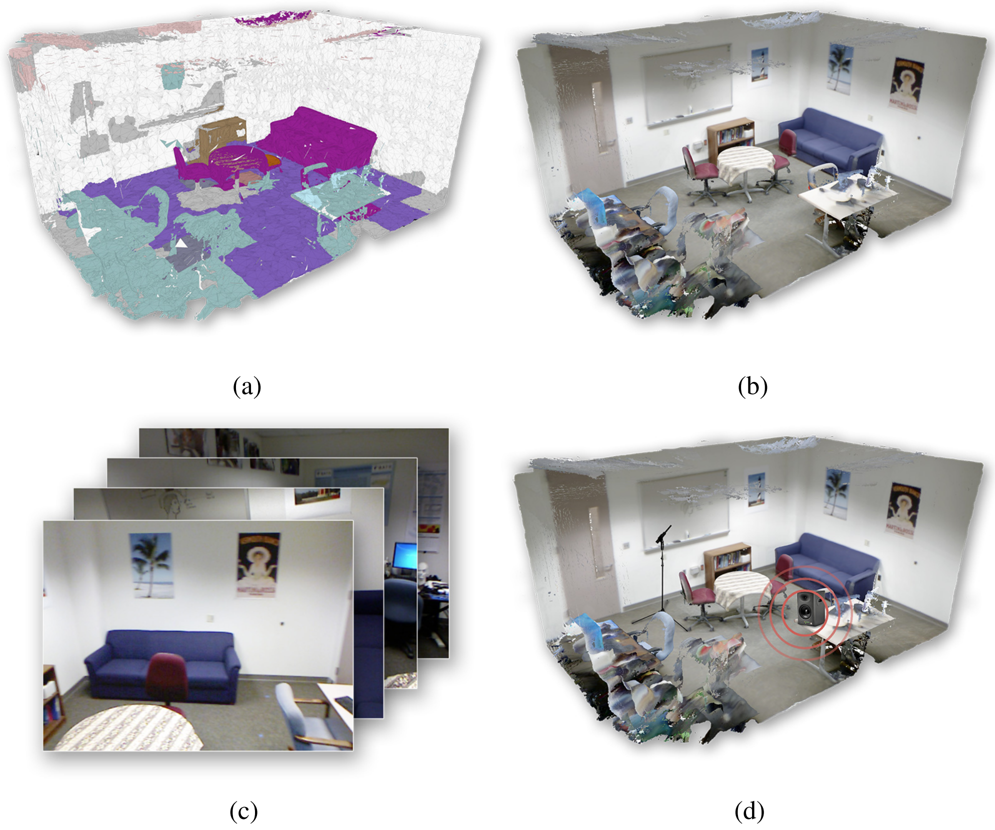


Figure 4.1: Our approach automatically estimates the acoustic materials, (a), of 3D reconstructions of real-world scenes, (b), using deep learning material classifiers applied to RGB camera images, (c). We optimize the material absorption coefficients to generate sound propagation effects that match acoustic measurements of the real-world scene using a simple microphone and speaker setup (d). The synthetic sound effects are combined with visual renderings of captured models for multimodal augmented reality.

4.2.1 Visual Material Classification for Acoustics

We present a new technique that uses the visual appearance of a real scene to estimate the acoustic material properties of the primitives. We make the assumption that there is a strong correspondence between the visual appearance of a surface and its acoustic material. For example, if a surface appears to be like brick, it is likely to have acoustic properties similar to the measured acoustic characteristics of a brick (e.g., to be highly reflective). The basis of our material classification approach is the *Materials in Context Database* (MINC) and its classifier models that have been trained for 23 common material categories (Bell et al., 2015). From these 23 categories, we select a subset of 14 that are likely to be encountered in real scenes and discard material categories that are less relevant for acoustic simulation (e.g. hair, skin, food). We manually associate each of the categories with measured data for similar acoustic material types from (Egan, 1988). For example,

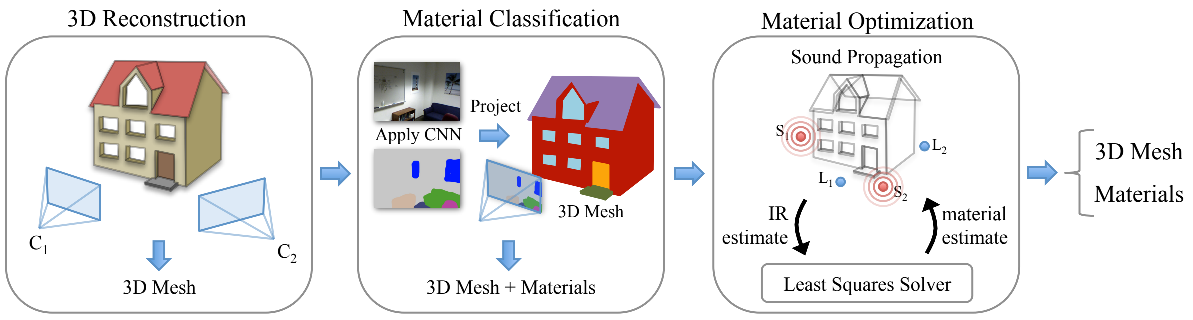


Figure 4.2: Our approach begins by generating a 3D reconstruction of a real-world scene from multiple camera viewpoints. Next, a visual material segmentation is performed on the camera images, producing a material classification for each triangle in the scene. Given a few acoustic measurements of the real scene, we use the visual materials as the initialization of our material optimization algorithm. The optimization step alternates between sound propagation simulation at the measurement locations and a material estimation phase until the simulation matches the measurements. The result is a 3D mesh with acoustic materials that can be used to perform plausible acoustic simulation for augmented reality.

the MINC "brick" material category is matched with the measured absorption coefficients for the "unpainted brick" acoustic material. When there is not a one-to-one mapping between the visual material categories and the acoustic material data, we pick the most similar acoustic material in the database. This process is performed once per material category. The resulting table of material categories and their associated acoustic materials are summarized in Table 4.1.

The MINC CNNs were trained using 3 million material patches from 436,749 images classified by human workers on Amazon Mechanical Turk. Bell et al. have shown that context, i.e. the image content surrounding a point of interest, is important in accurately classifying the materials in an image. For this reason, we choose to use images of real scenes as the input to the classification pipeline since they contain the necessary context information. For 3D scene reconstruction, a structured-light RGBD camera is used to capture the images of the scene. We use these images as the input to our material classification method. Using the approach of (Dou et al., 2013), we also generate a 3D triangle mesh for the scene with the color specified per-vertex. As part of the reconstruction, we assume that the camera projection matrix for each image is also available. These matrices are used to project the computed materials onto the mesh.

Our material classification approach is applied to each of the RGB camera images independently. We use a variant of the sliding-window approach detailed in (Bell et al., 2015) to apply the MINC trained GoogLeNet (Szegedy et al., 2015) to a grid of locations in each input image. The input to the network is a square image patch centered at the test location of size $p = d_{min} * 256/1100$ where d_{min} is the smaller of the image

		Absorption					
Visual Category	Acoustic Material	125Hz	250Hz	500Hz	1,000Hz	2,000Hz	4,000Hz
Brick	Brick, unglazed	0.02	0.02	0.03	0.04	0.05	0.07
Carpet	Carpet, heavy, on concrete	0.02	0.06	0.14	0.37	0.60	0.65
Ceramic	N/A	0.10	0.10	0.10	0.10	0.10	0.10
Fabric	Fabric Drapery, lightweight	0.03	0.04	0.11	0.17	0.24	0.35
Glass	Glass, ordinary window	0.35	0.25	0.18	0.12	0.07	0.04
Leather	Leather seating	0.44	0.54	0.60	0.62	0.58	0.50
Metal	Steel	0.05	0.10	0.10	0.10	0.07	0.02
Painted	Gypsum board, 1/2", with 4" airspace	0.29	0.10	0.05	0.04	0.07	0.09
Plastic	N/A	0.10	0.10	0.10	0.10	0.10	0.10
Stone	Concrete, rough	0.01	0.02	0.04	0.06	0.08	0.10
Stone, polished	Concrete, smooth	0.01	0.01	0.02	0.02	0.02	0.02
Tile	Tile, marble or glazed	0.01	0.01	0.01	0.01	0.02	0.02
Wallpaper	Gypsum board, 1/2", with 4" airspace	0.29	0.10	0.05	0.04	0.07	0.09
Wood	Wood, 1" panneling, with airspace	0.19	0.14	0.09	0.06	0.06	0.05

Table 4.1: The material categories and absorption coefficient data that was used in our classification approach. For each of the visual material categories, a similar acoustic material and its absorption coefficients were chosen from (Egan, 1988). For the "Ceramic" and "Plastic" categories, there was no suitable measured data available so a default absorption coefficient of 0.1 was assigned for all frequencies.

dimensions. The patches are extracted from the input image and scaled to 224×224 resolution. The mean of the patch is subtracted before it is passed through the CNN. At each test location, the CNN classifier predicts a probability for each material category. This grid of test locations is used to generate probability maps for all of the material categories. The probability maps are low-resolution images indicating the probability that a given material type is present at a position in the original image. The results are bilinearly filtered to the original image resolution and padded with zeros to maintain alignment before they are used to generate a final probability map for each camera image and material category.

Patch Segmentation: The next step is to determine the segmentation of material patches that should be used for the reconstructed 3D triangle mesh. These patches are localized groups of triangles that are assumed to be the same material. We use a 3D version of the SLIC superpixel segmentation algorithm (Achanta et al., 2012) and the vertex colors computed during reconstruction from the RGB images to determine the segmentation. In our particular implementation, we are concerned with clustering triangles rather than voxels, so we cluster according to the interpolated color of each triangle's centroid. The first step in the SLIC algorithm is to convert the RGB color for each triangle centroid to the LAB color space. Then, the initial superpixel cluster centers in 3D space are determined by sampling the bounding box of the mesh at regular interval s on a cubic grid. The sampling interval s is determined using the relation $s = (V/k)^{1/3}$, where V is the volume of the mesh's bounding box and k is the desired number of cluster centers. The initial color values for the cluster centers are chosen to be the colors of the nearest triangle centroids. Thus, each cluster and triangle is described by an (X, Y, Z, L, A, B) tuple.

Next, the SLIC algorithm iteratively refines the clusters until the maximum error for all clusters is lower than a threshold or until the algorithm converges. First, each cluster considers all triangles within a $2s \times 2s \times 2s$ region around the center point in 3D space. The distance in XYZLAB space between the triangle centroid and cluster center is computed according to the standard SLIC distance metric, and the cluster label for each triangle is chosen to be the cluster with the smallest distance. Then, the XYZLAB cluster centers are recomputed as the average of all triangle centroids that belong to a cluster. The residual error between the old and new cluster centers is determined using the L_2 norm in XYZLAB space. If the error is less than a threshold or if the error converges, the algorithm is terminated. The result is a collection of material patches that tend to closely match the visual features and boundaries in the reconstructed mesh. These patches are used as the basis of our material optimization algorithm (Section 4.2.2).

Material Projection: Next, the 2D classification results for all images are combined and applied to the reconstructed 3D triangle mesh. For each patch in the mesh, we create an accumulator p_i, initially set to zero, that stores the probability that the patch has the ith material type. Next, we project the material probabilities present in each image into the scene with the camera projection matrix. In our implementation, we perform this operation by tracing a ray for each image pixel. The patch intersected by the ray is updated by sampling from the probability map for the ith material type, and then we add the sampled probability to p_i. After this step has been carried out for every input image, we choose the final material for each patch to be the material with the largest p_i. By combining the results from many input images that are likely to have significant overlap, we achieve more robust material classification than could be achieved by using the results from a single image. Additionally, pooling the p_i for each material patch rather than for each triangle generates more robust material classifications that follow patch boundaries and are more likely to match the visual features of the mesh.

Mesh Simplification: The final step in preparing the reconstructed mesh for acoustic simulation is to simplify the dense triangle mesh. Dense 3D reconstructions frequently have triangles that are smaller than the smallest audible wavelength of 1.7cm, given by the speed of sound in the air and human hearing range. However,

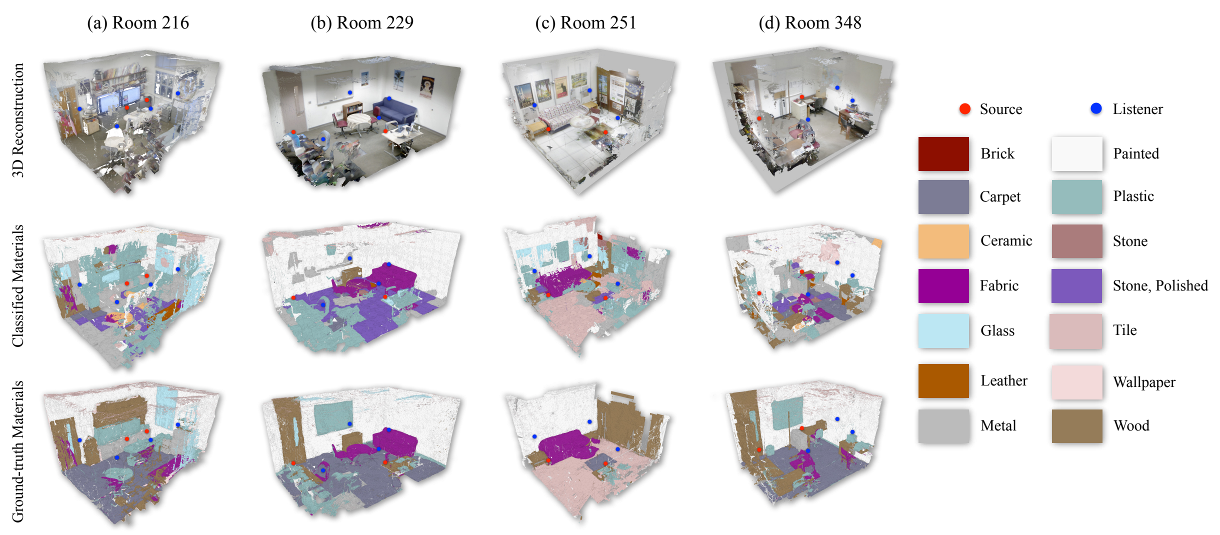


Figure 4.3: The results of our visual material classification algorithm for the four benchmark scenes. Colors indicate the material category that has been assigned to each triangle of the reconstructed model. The middle row shows the results of our material classification, and the bottom row shows the manually-generated ground-truth classification that are used for validation. The source and listener positions for the acoustic measurements within the real room are shown as red and blue circles, respectively. These are used to optimize the acoustic materials present in the scenes.

geometric sound propagation algorithms are generally more accurate when surface primitives are larger than audible sound wavelengths. Therefore, we apply acoustic mesh simplification techniques (Schissler et al., 2014) to the dense 3D mesh and its material properties to increase the size of surface primitives and to reduce the number of edges for diffraction computation. The simplification algorithm involves a combination of voxel remeshing, vertex welding, and the edge collapse algorithm to reduce the model complexity. Boundaries between the patches are respected by the simplification so that no additional error is introduced. This results is a mesh that is appropriate for geometric sound propagation.

4.2.2 Acoustic Material Optimization

While visual material classification algorithms can achieve good results for visually salient materials (e.g., brick and grass), other material types may be ambiguous (e.g., painted walls) or not included in the training set. Furthermore, the materials for occluded areas in the scene are also unknown. At the same time, these occluded areas contribute to the acoustic effects of reflections and diffraction. In addition, when applied to acoustic material classification, the techniques developed for visual materials do not consider non-visual properties like density and the presence of hidden resonant cavities in the scene that can also

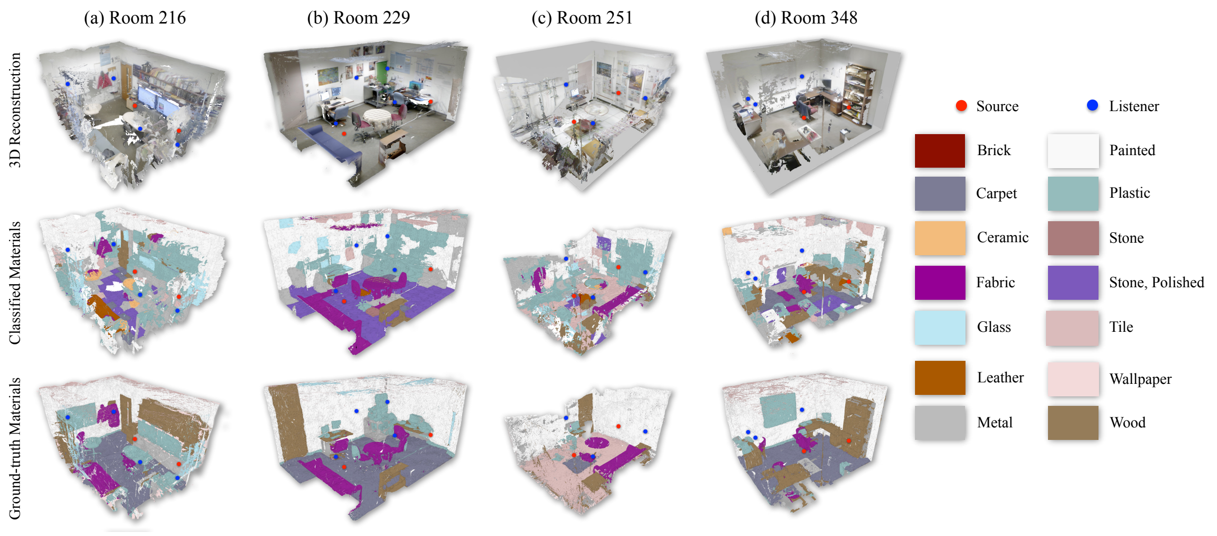


Figure 4.4: An alternate view of the results of our visual material classification algorithm for the four benchmark scenes.

affect the acoustic characteristics. The thickness and rigidity of walls also influences how sound propagates and these properties cannot be determined visually. As a result, a visual material classification algorithm used on the surfaces in a scene may not accurately classify the acoustic materials. Even if accurate material segmentation and classification information is known, the resulting sound simulated using that information may not match the real scene because the measured acoustic material data that is assigned to each material does not necessarily generalize to arbitrary scenes. Another issue is that holes in the 3D reconstructed mesh can cause the sound to 'leak' out of the scene, unnaturally decreasing the reverberation time. This problem can be mitigated by automatic hole-filling techniques (Wang and Oliveira, 2003; Branch et al., 2006), but they do not always produce a correct result and can introduce other meshing errors.

In order to overcome these issues, we utilize captured acoustic measurements in the real-world scenes. We propose a second pipeline stage that optimizes the visually-classified material properties, computed using the algorithm in Section 4.2.1, so that the resulting acoustic simulation more closely matches the IRs of acoustic measurements taken from the real-world scene. One simple possibility would be to use the reverberation time, RT_{60}, and Sabine reverberation equation to globally modify the absorption coefficients to match the measured RT_{60}. However, the Sabine model is only valid for rectangular rooms and does not consider other important quantities like the ratio of direct to late sound energy. The RT_{60} also doesn't vary much throughout an environment, and so it doesn't provide much information about the spatial locality of absorption. As a result, an approach based on matching only the RT$_{60}$ might lead to large errors with respect to other perceptually-relevant metrics.

Our formulation instead optimizes the sound intensity reflection coefficient R for each material patch and simulation frequency band using an iterative least-squares approach in order to minimize the error between intensity-time histograms from the simulation and intensity-time histograms from measured IRs. This is similar to the approach of (Saksela et al., 2015). However our technique improves on several significant limitations. Their method makes the assumptions that all reflections are specular, that there is no significant diffraction, that all sound propagation paths are discrete and known to the optimization system, and that there is a one-to-one correspondence between the paths in the optimized and target IRs. These assumptions can only be satisfied if the optimization target IR is computed using the same simulation that is used during optimization, which is not the case for measured IRs. In addition, the approach of (Saksela et al., 2015) only considers the early reflections computed via beam tracing and so it can't optimize the late reverberation present in real-world scenes that involves high-order diffuse reflections. These limitations prevent that method from optimizing acoustic materials to match real-world measurements.

Therefore, we introduce a new method that is able to handle the case of optimizing materials for sound rendering in real-world scenes.

Acoustic Measurements: The target of our optimization algorithm is a collection of impulse response measurements from the real scene. For each IR measurement, there is a corresponding source and listener placed within the virtual reconstructed 3D mesh. The target measured IR for a single source/listener pair is given by the time-domain signal $p^T(t)$, while the IR computed in the virtual scene for the same source/listener pair is given by the signal $p^S(t)$. To use these pressure IRs in our optimization approach, the first step is to filter them into the frequency bands used for the sound propagation simulation. This yields $p^T_{\bar{\omega}}(t)$ and $p^S_{\bar{\omega}}(t)$ for frequency band $\bar{\omega}$. Then, the Hilbert Transform is applied to extract the pressure magnitude envelope from the filtered IRs (Bendat and Piersol, 1986). The square of the IR envelope then yields the intensity-time curve for each impulse response, indicating the distribution of sound intensity over time. The intensity-time curve for the target and simulated impulse responses and frequency band $\bar{\omega}$ are given by $I^T_{\bar{\omega}}(t)$ and $I^S_{\bar{\omega}}(t)$ respectively. The high-level goal of the optimization algorithm is to minimize the error between $I^S_{\bar{\omega}}(t)$ and $I^T_{\bar{\omega}}(t)$. Since the number of time samples in the IRs may be on the order of 10^5, it is necessary to perform the optimization at a lower sampling rate than the audio rendering sample rate to reduce the size of the optimization problem and increase its robustness. This is done by binning the intensity present in the intensity-time curves to produce intensity-time histograms $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$, where b is the bin index. Thus, our algorithm in practice minimizes the error between $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$. The intensity for bin b in each IR is given by: $I^S_{\bar{\omega}b} = \sum_{t_b \in b} I^S_{\bar{\omega}}(t_b)$ and $I^T_{\bar{\omega}b} = \sum_{t_b \in b} I^T_{\bar{\omega}}(t_b)$. The bin size L is a parameter that determines the time resolution of the optimization and it impacts the robustness, convergence, and performance. We used $L = 10$ms.

IR Registration: On each iteration of our optimization algorithm, the simulated IR must be registered with the measured target IR so that it has the same time alignment and similar amplitude. This is very important for correct operation of our algorithm. If bins $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$ do not correspond to the same time window in the IR, then the error between them can be very large and this can lead to incorrect results as the error grows on subsequent iterations. To rectify this, we propose a method for registering the IRs that is robust to the presence of noise in the measured IR. The registration operation is performed independently for every frequency band and at each optimization iteration. The first step is to compute the cross correlation between the IRs at every time offset. The simulated IR is then shifted in time to the offset where the cross correlation is highest. Once the IRs have been time aligned, the amplitudes must be matched. A significant problem with matching them robustly is that the signal-to-noise ratio (SNR) of the measured IR may be poor due to the presence of ambient noise. This noise produces incorrect registration which can lead to poor optimization performance. As a result, we only consider the bins in the IRs that have intensity over the noise floor for both IRs. Given a signal-to-noise ratio for each IR, SNR^T and SNR^S, we determine the noise floors to be $\epsilon^T = \frac{\max(I^T_{\bar{\omega}}(t))}{\text{SNR}^T}$ and $\epsilon^S = \frac{\max(I^S_{\bar{\omega}}(t))}{\text{SNR}^S}$. In the case of our measurement data, SNR$^T \approx 10^4$ and SNR$^S = \infty$. Then, an intensity scale factor λ for the simulated IR that minimizes the L_2 error between all bins $I^T_{\bar{\omega}b} > \epsilon^T$ and $I^S_{\bar{\omega}b} > \epsilon^S$ is computed using a least-squares solver. $I^S_{\bar{\omega}b}$ is multiplied by λ to yield a simulated IR that is registered to the target measured IR. The registered IRs are used on each iteration of our algorithm to estimate the error for each IR bin. The error in decibels for bin b between the simulated and target IRs is given by $E_{\bar{\omega}b} = \text{dB}(I^T_{\bar{\omega}b}) - \text{dB}(I^S_{\bar{\omega}b})$ where $\text{dB}(x) = 10\log_{10}(\frac{x}{I_0})$ and I_0 is the reference sound intensity.

Acoustic Simulation: A key part of our algorithm is the incorporation of sound transport information from virtual simulations within the scene's 3D reconstruction. We use a ray-based geometric sound propagation system that computes $I^S_{\bar{\omega}}(t)$ directly as the sum of many individual ray paths, e.g. $I^S_{\bar{\omega}}(t) = \sum \delta(t - t_j) I_{j\bar{\omega}}$

where $I_{j\bar{\omega}}$ is the sound intensity for path j and frequency band $\bar{\omega}$, t_j is the propagation delay time for path j, and $\delta(x)$ is the Dirac delta function. Along with $I^S_{\bar{\omega}}(t)$, the sound propagation system also computes a *weight matrix*, $W_{\bar{\omega}}$, for each frequency band. $W_{\bar{\omega}}$ has rows corresponding to the impulse response bins and columns corresponding to the material patches present in the scene. For IR bin b and patch m, the entry of $W_{\bar{\omega}}$ is given by $w_{\bar{\omega}bm} = \frac{\sum I_{j\bar{\omega}} d_{jm}}{\sum I_{j\bar{\omega}}}$ where d_{jm} is the number of times that path j hit material patch m during its scene traversal. Therefore, $w_{\bar{\omega}bm}$ represents the average number of reflections involving patch m for all paths that arrived at the listener during bin b, weighted according to the sound intensity of each path. Essentially, $W_{\bar{\omega}}$ encodes the amount of influence each material patch has on every IR bin. The weight matrix is used during the optimization procedure to estimate the best changes to make to the material patches to minimize the error between $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$.

Solver System: In the unlikely case where there is just one sound propagation path per bin in the impulse response, the intensity for a simulated impulse response bin is given by:

$$I^S_{\bar{\omega}b} = \frac{P}{4\pi N} \prod_d R_{j\bar{\omega}d} \tag{4.1}$$

where P is the sound source's power, N is the number of primary rays emitted from the source, and $R_{j\bar{\omega}d}$ is the frequency-dependent intensity reflection coefficient encountered along path j at reflection bounce d. Converting $I^S_{\bar{\omega}b}$ to decibels by taking the logarithm allows the intensity to be expressed as a linear combination of the logarithm of reflection coefficients:

$$\mathrm{dB}(I^S_{\bar{\omega}b}) = \mathrm{dB}\left(\frac{P}{4\pi N}\right) + \sum_d \mathrm{dB}\left(R_{j\bar{\omega}d}\right). \tag{4.2}$$

In the approach of (Saksela et al., 2015), this relationship is used to directly solve for the reflection coefficients that produce $\mathrm{dB}(I^S_{\bar{\omega}b}) \approx \mathrm{dB}(I^T_{\bar{\omega}b})$ in a least-squares sense. Given the weight matrix $W_{\bar{\omega}}$ obtained during the simulation, the system of equations solved by (Saksela et al., 2015) is roughly:

$$W_{\bar{\omega}}\mathrm{dB}(R_{\bar{\omega}}) = \mathrm{dB}(I^T_{\bar{\omega}}) - \mathrm{dB}\left(\frac{P}{4\pi N}\right) \tag{4.3}$$

where $R_{\bar{\omega}}$ is a vector of the reflection coefficients for each material patch, and $I^T_{\bar{\omega}}$ is a vector of the intensity for the bins of the target impulse response. After solving for $\mathrm{dB}(R_{\bar{\omega}})$, the reflection coefficients can be directly determined by inverting the decibel transform. This formulation requires a one-to-one correspondence between the propagation paths in the simulated and target impulse responses and so cannot be used in the presence of diffuse reflections or diffraction because these phenomena introduce scattering that "blurs" the boundaries between paths. Their approach also requires accounting explicitly for the effects of additional acoustic phenomena such as air absorption. In addition, since it is difficult to extract discrete paths from a measured impulse response, especially for late reverberation, the technique of (Saksela et al., 2015) cannot be applied to real-world measurements.

To handle these problematic cases, we reformulate the optimization problem as an approximate iterative algorithm. In the general case where many paths are assigned to the same bin, the intensity in each bin of the simulated IR is given by:

$$I^S_{\bar{\omega}b} = \frac{P}{4\pi N} \sum_j \prod_d R_{j\bar{\omega}d}. \tag{4.4}$$

This produces a non-linear system of equations that is difficult to handle accurately within the framework of (Saksela et al., 2015). Therefore we make the assumption that most of the paths in the same bin will hit a similar number of patches during the scene traversal. While this assumption introduces some error, it allows the use of a similar least-squares formulation. Rather than solving directly for the reflection coefficients, we instead iteratively estimate the change in decibels to each reflection coefficient that minimizes the error between the simulated and target IRs. This enables our algorithm to automatically handle a wider range of acoustic phenomena and means that it is robust to external influence from noise. The system solved on each iteration is given by:

$$W_{\bar{\omega}} \mathrm{dB}(\Delta R_{\bar{\omega}}) = E_{\bar{\omega}} \tag{4.5}$$

where $\mathrm{dB}(\Delta R_{\bar{\omega}})$ is a vector of the change in decibels for each patch's reflection coefficient, and $E_{\bar{\omega}}$ is a vector of the error between the simulated and target IRs in decibels for all IR bins. The reflection coefficients are then updated on each iteration by $R'_{\bar{\omega}m} = R_{\bar{\omega}m} \Delta R_{\bar{\omega}m}$. To enforce physical plausibility, the reflection coefficient should be constrained to the range $[R_{min}, R_{max}]$ encountered for typical real-world materials. We use $R_{min} = 0.3$ and $R_{max} = 0.999$. For other applications (e.g. architectural acoustics) it may be useful to apply additional constraints on material placement.

This approach can be easily extended to handle the case where there are multiple measured IRs. If $W^i_{\bar{\omega}}$ is the weight matrix computed for IR i, then the final weight matrix $W_{\bar{\omega}}$ used to solve the system for all IRs is formed by vertically concatenating the rows of each $W^i_{\bar{\omega}}$. Similarly, if $E^i_{\bar{\omega}}$ is the error in decibels for IR i, then the final error vector $E_{\bar{\omega}}$ is the vertical concatenation of the various $E^i_{\bar{\omega}}$. The final optimized materials will then incorporate information from every measured IR.

Optimization: The optimization begins with the initial materials for every patch in the scene as determined in Section 4.2.1. Then, our iterative constrained least-squares optimization algorithm is applied to modify the materials so that the simulation better matches the real scene. The main steps of our algorithm at each iteration are summarized below:

1. For each IR in the scene, build the solver system:
   (a) Compute simulated intensity-time curve $I^S_{\bar{\omega}}(t)$ and weight matrix $W_{\bar{\omega}}$.
   (b) Register simulated IR $I^S_{\bar{\omega}}(t)$ to target measured IR $I^T_{\bar{\omega}}(t)$.
   (c) Bin $I^S_{\bar{\omega}}(t)$ and $I^T_{\bar{\omega}}(t)$ into intensity-time histograms $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$ with bin size L.
   (d) Compute, $E_{\bar{\omega}b}$, the error in decibels between $I^S_{\bar{\omega}b}$ and $I^T_{\bar{\omega}b}$.
2. Solve least-squares system to get change in reflection coefficients, $\Delta R_{\bar{\omega}m}$.
3. Apply $\Delta R_{\bar{\omega}m}$ to material patches $R_{\bar{\omega}m}$, enforcing constraints.
4. Check termination conditions.

The algorithm terminates once a maximum number of iterations has elapsed, if the average per-bin error is less than 1 decibel, or the algorithm converges to a local minimum.

4.3 Implementation

In this section, we describe the implementation of various components of our acoustic material optimization and classification approach.

3D Model Reconstruction: We generated a 3D reconstruction of each scene (i.e., a real-world room) using a few hundred RGB-D images captured with a Microsoft Kinect at 640x480 resolution. The reconstruction algorithm utilizes high-level plane primitive constraints and SIFT features (Dou et al., 2013) to compute the 3D triangle mesh that is used as an input by our acoustic classification and optimization algorithm. The captured images are also used as the input to our classification algorithm. All material classifications were performed using the Caffe deep learning framework (Jia et al., 2014) and MINC patch classifier models (Bell et al., 2015) on an Nvidia GTX 960.

Sound Propagation and Rendering: For computation of sound propagation, we use a combination of the image source method for specular early reflections (Borish, 1984), diffuse path tracing for late reverberation (Vorländer, 1989; Schissler et al., 2014), and the UTD diffraction model for approximating edge diffraction (Tsingos et al., 2001). All the sounds in our system are computed in 8 octave frequency bands centered at 63.5Hz, 125Hz, 250Hz, 500Hz, 1kHz, 2kHz, 4kHz, and 8kHz. The paths computed by the underlying propagation algorithm are used in the material optimization algorithm. The result of sound propagation is an impulse response for every source and listener pair in the scene. The impulse response is spatialized using vector-based amplitude panning (Pulkki et al., 2001) and the unprocessed anechoic source audio is convolved with the IR. We also compute perceptually important sound paths such as the direct and early reflection paths separately from the impulse response so that linear delay interpolation (Wenzel et al., 2000) and HRTF spatial sound can be applied independently for each path. The outputs of delay interpolation and convolution rendering are mixed and then sent to the audio device for playback over headphones.

Acoustic Measurements: The ground-truth acoustic measurements for our optimization approach consist of impulse responses at various source and listener locations in the real scenes. The measurements are captured using complimentary Golay codes (Foster, 1986) played through a JBL LSR4328P speaker and captured by a Beyerdynamic MM1 omnidirectional measurement microphone. The measurement setup is depicted in Figure 4.1 (d). These measurements for each room take about $20 - 30$ minutes, including the time to setup the equipment. We measured 4 impulse responses for each scene that correspond to 4 different speaker/microphone pairs. Each IR was measured 20 separate times and then the results were averaged to reduce the amount of noise present. These measured IRs are used for auralization and are also used during our optimization algorithm. The positions and orientations of the microphone and speaker with respect to the 3D reconstruction were also measured and entered into the simulation. In order to correctly replicate the directional characteristics of the speaker in the sound propagation simulation, we also measured the directional transfer function of the speaker in free space at 15 azimuths and 4 elevations for a total of 60 measurements. The magnitude response of the transfer function in the 8 octave frequency bands is used to model the directional speaker in our simulation so that there is better correspondence with the measured IRs. Overall, the audio capture requirements of our approach are low and robust to measurement errors from consumer audio equipment (e.g., with nonlinear microphone or speaker frequency response). In situations

	Scene Complexity			Classification			Optimization
Scene	Dimensions (m)	# Triangles	RT_{60} (s)	# Images	Time (hr)	% Correct	Time (s)
Room 216	5x3.8x2.6	228,017	0.29	486	9.3	43.6	142.2
Room 229	5.3x3.6x2.6	139,545	0.44	481	9.2	52.5	154.1
Room 251	3.9x3.3x2.6	168,098	0.37	340	6.5	43.8	134.9
Room 348	4.3x3.1x2.6	131,194	0.37	481	9.2	51.4	150.5

Table 4.2: This table provides the details of the four room-sized benchmark scenarios. We give the physical dimensions of each room and the geometric complexity of the 3D reconstructed mesh models after simplification, as well as the RT$_{60}$ values computed from the IRs. We highlight the time spent in the material classification and the optimization algorithms, as well as the percentage of scene surface area correctly classified.

with low background noise, the impulse response can even be estimated by recording the result of the user clapping their hands.

4.4 Results

We have evaluated our acoustic material classification and optimization approach on several indoor real-world scenes typical of an office environment. The major characteristics and results for these scenes are summarized in Table 4.2. Our approach is not optimized and currently takes 6-9 hours to classify the materials in each scene. The results for the visual material classification are shown in Figure 4.3. We compare the output of the classification approach to a manually segmented mesh that is used as the ground truth. Overall, about 48% of the triangles in the scenes are correctly classified. For some of the materials that are incorrectly labeled (e.g. the carpet floor in Room 229 labeled as “Stone, polished”), it is possible that the CNN is unable to tell the difference between certain types of visually similar materials. A possible explanation for these results is that the input image resolution used in our implementation is less than half of the training images for the MINC dataset (1100px vertical dimension), and the RGB images contain lots of noise. There is not enough salient high-frequency information in the input images for the CNN to detect all of the material types. Another shortcoming of our automatic acoustic material classification is that the CNN cannot accurately predict materials that require higher-level knowledge of the scene. For instance, some of the test rooms contain posters on the wall that have negligible effect on the acoustics, and so the wall material (e.g. “painted”) should be used. However, the CNN can be confused in these situations and produces incorrect predictions that cannot be rectified without high-level knowledge of how sound propagates through materials.

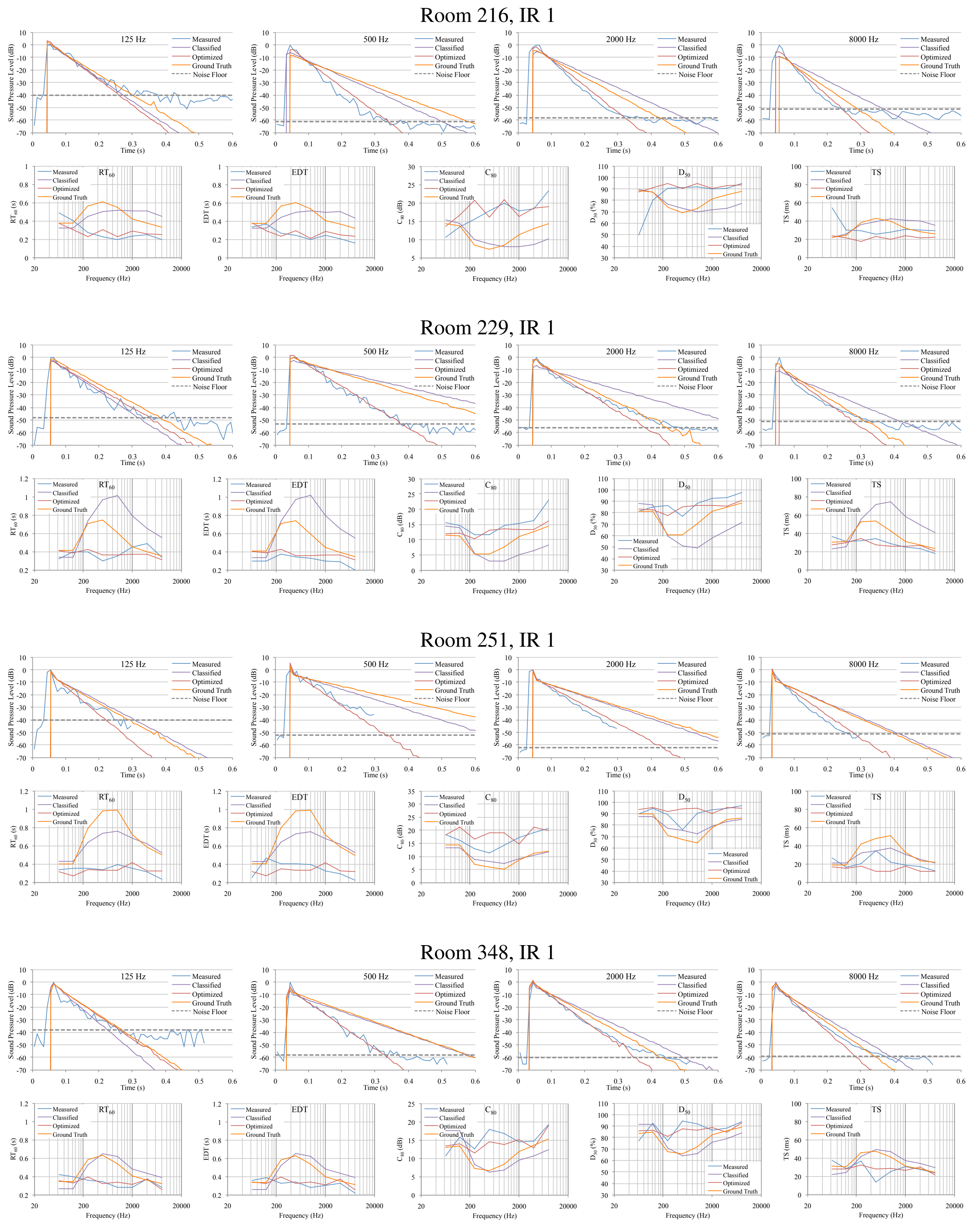
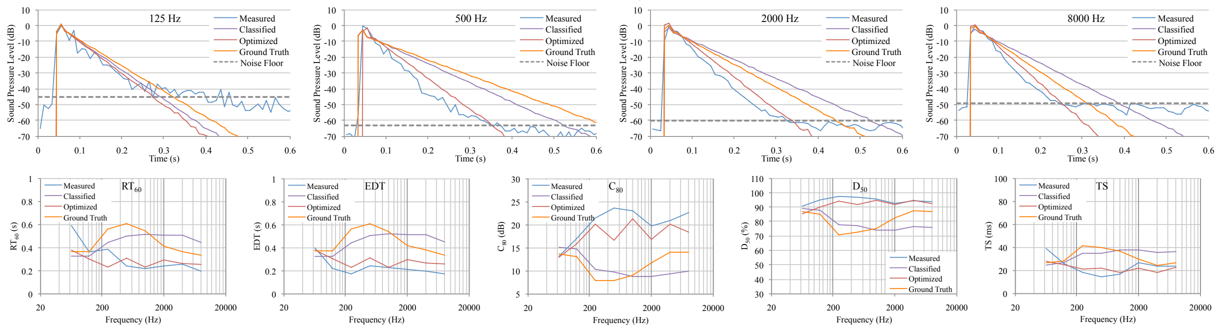
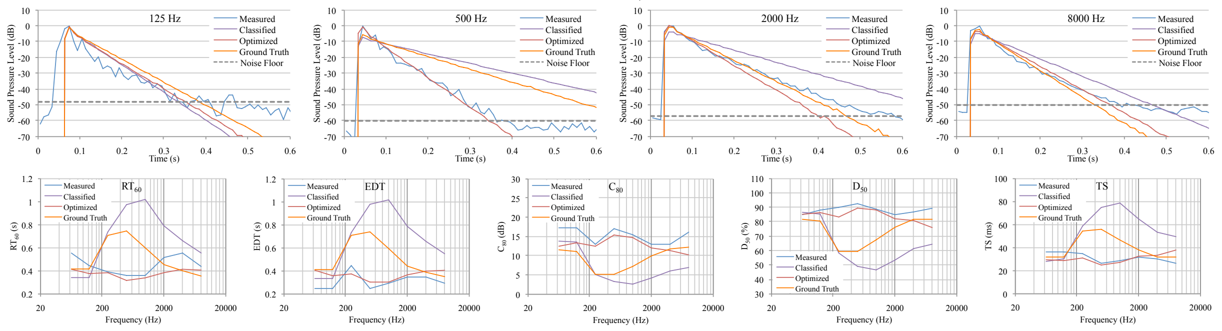


Figure 4.5: The main results of our material optimization approach. We compare the energy-time curves and several standard acoustic parameters for the measured IRs (*measured*) to the results before optimization (*classified*) and the optimized results (*optimized*). We also show the results for manually-segmented materials without optimization (*ground truth*). The energy-time curves are presented for four octave frequency bands with center frequencies 125Hz, 500Hz, 2000Hz, and 8000Hz. The noise floor corresponds to the signal to noise ratio of the measured IR for each frequency band.

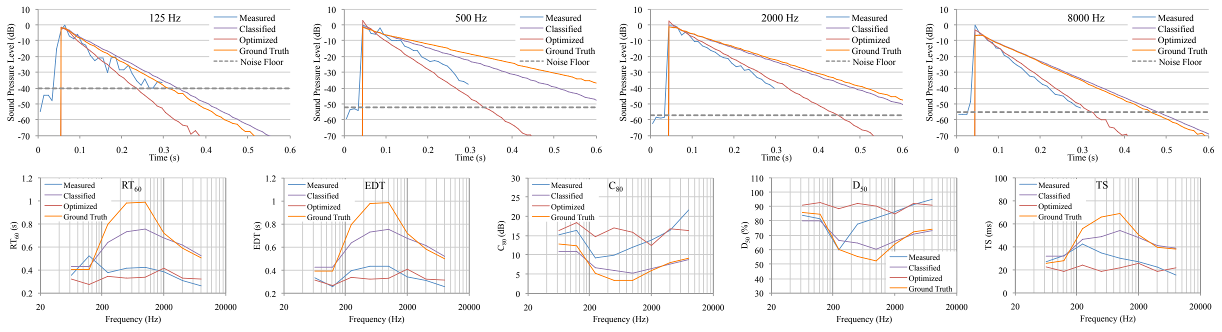
Room 216, IR 2



Room 229, IR 2



Room 251, IR 2



Room 348, IR 2

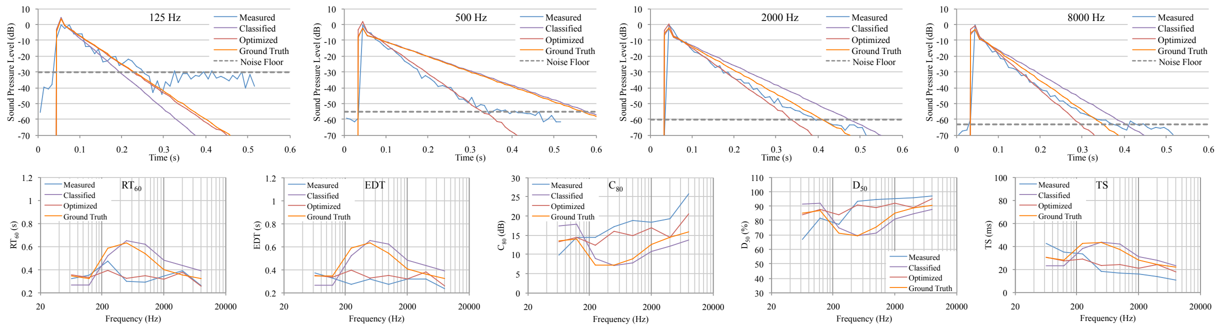


Figure 4.6: Results of our optimization approach for alternate impulse responses. We compare the energy-time curves and several standard acoustic parameters for the measured IRs (*measured*) to the results before optimization (*classified*) and the optimized results (*optimized*). We also show the results for manually-segmented materials without optimization (*ground truth*). The energy-time curves are presented for four octave frequency bands with center frequencies 125Hz, 500Hz, 2000Hz, and 8000Hz. The noise floor corresponds to the signal to noise ratio of the measured IR for each frequency band.

On the other hand, our acoustic material optimization algorithm for computation of absorption coefficients improves the accuracy of the simulated impulse responses with respect to the measured IRs. The results of our optimization algorithm for one IR in each scene are shown in Figure 4.5. Results for additional additional IRs can be found in the supplementary document. We show the energy-time histograms for the 125Hz, 500Hz, $2,000$Hz and $8,000$Hz frequency bands. For each band, we compare the measured data to the results produced directly after material classification, as well as the final results generated by our optimization algorithm. Before optimization, there are several significant mismatches with the measured data, especially in the mid-frequency bands. This can be partially explained by error during classification, though another significant factor is the table of measured absorption coefficients (Table 4.1), which may not always be valid for general scenes with various wall construction techniques. When the classified results are compared to the ground truth materials (before optimization), we observe similar error for both with respect to the measured data. This supports the conclusion that the material database does not match the test rooms well.

After optimization is applied, the impulse responses are much closer to the measured data. Overall, the optimization results are very close for the mid frequencies ($500 - 1000$Hz), with an average error of just $1 - 2$dB. At low frequencies there is a substantial amount of noise in the measured data, and the energy decay is not very smooth. This causes some error when compared to the optimized results that have a perfectly smooth energy decay curve. At high frequencies, we observe that the measured IRs tend to decay at a decreasing rate over time (e.g. not linear in decibels). This is most visible for Room 229 in the 8kHz frequency band. Our simulation does not reproduce this effect and it generates completely linear decay curves. This is an additional source of error that may require more sophisticated sound propagation algorithms to rectify. Noise in the measured IRs also makes it difficult to precisely match the energy decay for the entire impulse response decay. If the noise floor is set too high, then not all of the IR is considered during optimization. If the noise floor is set too low, such that the optimization considers the noise to be part of the IR, then it tends to produce a result with a slower incorrect decay rate. Setting the signal to noise ratio for each IR and frequency band is important for correct operation of the algorithm.

We also compared the results for several standard acoustic parameters: reverberation time (RT$_{60}$), early decay time (EDT), clarity (C$_{80}$), definition (D$_{50}$), and center time (TS). For the RT$_{60}$ and EDT, the optimized results are close to the measured results for most frequencies, and the average error is $0.08s$. There are some mismatches at low frequencies, but these are explained by noise at the end of the IR incorrectly increasing the reported decay rates for RT$_{60}$. The EDT parameter is less sensitive to this noise because it only considers the first 10dB of energy decay. The C_{80}, D_{50}, and TS parameters consider the ratio of early to late sound in different ways. The results for these parameters are mixed. In some cases the optimized parameters are very close to the measured data (e.g. 1dB), but for others there are significant differences (e.g. over 5dB). A few errors are caused by noise impacting the computation of the parameters. In particular, for the computation of C_{80} and D_{50}, it is important to know the time of first arrival in the IR. If this time is not determined accurately, it can cause significant differences in the reported parameters. Other errors seem to be mostly random, so it is hard to tell what is the cause. Overall, our optimization algorithm is able to match the measured data reasonably well, though there may be room for future improvement.

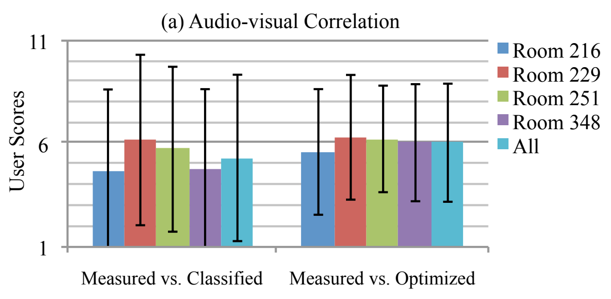
Analysis: The accuracy of our approach depends on four main components: the quality of the reconstructed 3D mesh model, the resolution of the input images and machine learning approach used for material classification, the database of acoustic materials, and the sound propagation model. While we may not need to reconstruct some small features of the scene (e.g. the door knob or a book on the shelf), it is important that we get nearly watertight meshes with only small holes, otherwise the sound waves can ‘leak’ from those models. Furthermore, the reconstruction algorithm may not capture some hidden areas that affect the sound propagation characteristics (e.g., hidden resonant cavities or non-line-of-sight large objects). It is important to acquire high resolution input images of the scene, as that affects the accuracy of the classification algorithm. Ideally, we want to use classifier models that are trained in terms of acoustic material categories and take into account the relationship between the visual appearance and the acoustic properties, but such data does not exist. As more acoustic material databases are becoming available, we can use them to increase the fidelity of our material classification algorithm. Finally, our optimization approach is based on geometric sound propagation which may not accurately simulate all sound phenomena. As a result, the optimization may not produce impulse responses that are identical to the measured ones. Ultimately, we would like to use more accurate models for sound propagation such as wave-based methods like the Boundary Element Method, but that would increase the computational complexity of our optimization algorithm substantially. Interestingly, all these four components are active areas of research in different research communities: computer vision, learning, acoustics, and scientific computation.

Applications: The proposed technique has many possible applications where it is useful to generate physically-based sound effects for real-world scenes. On such application is teleconferencing. When a remote person is speaking, the sound from their voice can be auralized as if they are within the room. By combining our technique with head or face tracking, it may also be possible to estimate the IR between a human voice and microphone. This IR can be deconvolved with the microphone audio to approximate the anechoic sound from the voice. Another application is to use virtual sound sources to provide feedback for an augmented reality user interface. For example, a virtual character that is overlaid onto the real world could communicate with the user. Our technique would allow the character's voice to be auralized as if it was within the real environment. The audio could be presented through open-back headphones that produce virtual sound with little attenuation of the external sound.

4.5 User Evaluation

In this section, we present results from a preliminary user study that evaluates the perceptual plausibility of our acoustic classification and optimization algorithms in terms of generating plausible sounds in real-world scenes.

Study Design: In the study, we compare sound auralized using measured impulse responses, referred to as *measured*, to the sound simulated using our technique both before and after absorption coefficient optimization, referred to as *classified* and *optimized*, respectively. We evaluated 2 comparison conditions: *measured* vs. *classified* and *measured* vs. *optimized*. For each case, we tested 2 source and listener pairs for each of the 4 scenes, for a total of 16 comparisons. The study was conducted as an online survey where each subject was presented with a 16 pairs of identical videos with different audio in a random order and asked two questions about each pair. The questions were (a) "which video has audio that more closely matches the visual appearance of the scene?" and (b) "how different is the audio in the videos?". The responses were recorded on a scale from 1 to 11. For the first question, an answer of 1 indicates the audio from the left video matched the visuals better, an answer of 11 indicates the audio from the right video matched the visuals better, and an answer of 6 indicates the videos were equal. On the second question, an answer of 1 indicates the videos sounded extremely similar, while an answer of 11 indicates the videos sounded very different. Our research hypotheses were: (1) the *optimized* case has sound with the same level of audio-visual correlation as the *measured* case and better correlation than the *classified* case; (2) The sound generated in the *optimized*



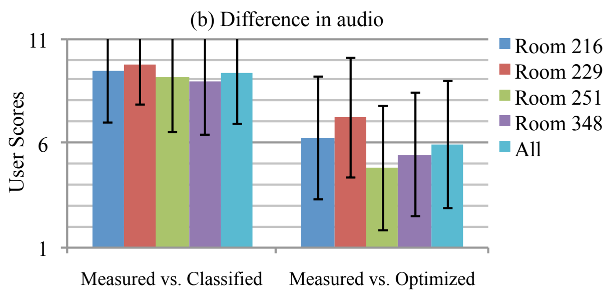


Figure 4.7: We compare the user evaluation results for *measured* case versus the *classified* and *optimized* cases for 4 scenes and 2 questions. For question (a), a score below 6 indicates higher audio-visual correlation for the first method in the comparison, whereas a score above 6 indicates higher audio-visual correlation for the second method. For question (b), the higher the score, the more dissimilar the audio for the two cases under comparison. Error bars indicate the standard deviation of the responses.

case will be more similar to that from the *measured* case than the sound from the *classified* case.

Study Procedure: The study was completed by a total of 19 subjects between the ages of 18 and 61, made up of 17 males and 2 females. The average age of the subjects was 28, and all subjects had normal hearing. At the start of the study, subjects were given detailed instructions and filled out a demographic information questionnaire. Subjects were required to use either headphones or earbuds when taking the study, and they were asked to calibrate their listening volume using a test audio clip. Subjects were then presented the 16 pairs of videos and responded to the questions for each pair. The subjects were allowed to replay the videos as many times as needed, and they were able to move forward and backward in the study to change their answers if necessary. After rating the 16 video pairs, the study was completed.

Study Results: The main results of our user evaluation are summarized in Figure 4.7. A two-tailed one-sample t-test across all scenes was used to test hypothesis 1. For question (a), the subjects indicated the video with the audio that more closely matched the visual appearance of the scene. For the comparison between the *measured* case and the *classified* case, the *measured* case is slightly preferred ($p = 0.038$), with an average score of 5.3 across the scenes. This indicates that the raw output of our classification approach does not match the visual appearance of the scene very closely. However, the comparison between the *measured* and *optimized* cases indicates that there is no preference for either case ($p = 0.82$), with an average score of 6.1. The low significance supports our first hypothesis that the level of audio-visual correlation for the *measured* and *optimized* cases is similar. Therefore, our approach is suitable for augmented reality applications that require generating virtual audio that matches the appearance of the real-world scene.

For question (b), the audio differences between the 3 cases were considered. When comparing the audio from the *measured* and *classified* case, the average user score was 9.3, suggesting strong differences in the audio. On the other hand, the audio from the *optimized* case was more similar to the measured audio, with an average user score of 5.9. When the second hypothesis is evaluated using a two-tailed Welch's t-test, we find that the *optimized* sound has much fewer differences as compared to the *measured* audio than the sound without optimization ($p < 0.001$). This suggests that the optimization step is important for generating sound that is close to the real-world scene.

Overall, the responses of the subjects varied significantly across individuals, producing large standard deviations. Some subjects could reliably tell the difference between the sound conditions, but other subjects seemed to be guessing, especially for the question concerning audio-visual correlation. The inclusion of more subjects and expert listeners could improve the quality of these results.

4.6 Conclusion, Limitations, and Future Work

We have presented a novel technique for acoustic material classification and optimization for 3D reconstructions of real-world scenes. Our approach uses a CNN classifier to predict the material categories for 3D mesh triangles, then iteratively adjusts the reflection coefficients of the materials until simulated impulse responses match corresponding measured impulse responses. We evaluated this technique on several room-sized real-world scenes and demonstrated that it can automatically generate acoustic material properties and plausible sound propagation effects. We used the results for multimodal augmented reality that combines real-world visual rendering with acoustic effects generated using sound propagation. Our initial results are promising and we also conducted a preliminary user study that suggests that our simulated results are indistinguishable from the measured data.

Limitations and Future Work: Our approach has some limitations. The accuracy of our approach is governed by the sensor resolution and underlying 3D model reconstruction algorithms. The input images for our material classification system have low resolution and our approach may not work well in this case. Moreover, the current approach of assigning measured material data to the MINC material categories can produce incorrect results. The number of categories is small and therefore the current MINC CNN model doesn't handle all real-world material variation. However, the material optimization technique proposed in Section 4.2.2 can be used to adjust the absorption coefficients so that the simulation is more consistent with acoustic measurements from the real scene. It is possible that the optimization may not converge to physically-accurate absorption coefficients because our approach may get stuck in local minima. Our technique also may not work well in scenes with many dynamic objects (e.g. humans) that can affect the sound. However, the impact of these objects is usually negligible if they are small in proportion to the size of the scene.

There are many avenues for future work in this domain. Most work in computer vision has been targeted towards 3D model reconstruction for visual rendering, and we need different criteria and techniques for sound rendering. Similarly, there is lack of measured data corresponding to acoustic-BRDFs for most real-world materials. It would be useful to extend recent work on visual material property acquisition (Weinmann and Klein, 2015) to acoustic materials. There is not always a one-to-one mapping between the visual material categories and the acoustic material data. This can be improved by training CNN models for new material types that disambiguate between specific acoustic material categories (e.g. painted vs. unpainted brick). Our approach also lacks high-level knowledge of the scene and materials, and so can produce incorrect material predictions where high-level knowledge is needed. Introducing additional material categories or features, such as the surface normal or size of the room, may help to classify problematic materials. Another possible avenue for improvement would be to try alternative machine learning models such as the support vector machine (SVM) or genetic algorithms. For instance, an SVM could be used to classify materials based on features extracted by the CNN (Tang, 2013). We have only considered the effects of the absorption/reflection coefficient on the impulse response. It may be possible to achieve better results by simultaneously optimizing for other material attributes like the scattering coefficient s, or by considering acoustic metrics like RT_{60}, the early decay time (*EDT*), or clarity (C_{80}) as constraints. We would also like to further evaluate our technique on larger indoor scenes with varying reverberation effects (e.g. cathedrals, concert halls), as well as outdoor scenes.

CHAPTER 5: LOW LATENCY SPATIAL SOUND FOR SOUND PROPAGATION [1]

5.1 Introduction

The computation of low-latency spatial sound is an important component of generating plausible sound propagation within virtual interactive environments. The goal of spatial sound is to reproduce the differences in sound heard at each ear by filtering the left and right channels according to the direction of sound arrival. This gives the user the sensation that the sound source is localized at a particular position in 3D space. Generating spatial sound for point sound sources is a well-studied problem (Møller, 1992). However, there are significant challenges in efficiently rendering spatial sound for area sources and in combining spatial sound with sound propagation. Existing approaches are not suitable for interactive applications because they can take several hundreds of milliseconds to update the spatial sound for a single large area source or a typical sound propagation impulse response (Lentz et al., 2007). This latency causes a noticeable delay when the listener's head is rotated and detracts significantly from the interactive audio experience (Brungart et al., 2004, 2005). Psychoacoustic research has suggested that the total acceptable latency is around 100ms (Lindau, 2009).

In this chapter, we present techniques for efficient low-latency computation of spatial sound for sound propagation. First, we describe an approach for efficiently computing spatial sound filters for sources that are represented by an area or volume (Section 5.3). Our approach computes the incoming sound field from an area source in the spherical harmonic (SH) basis functions using either an analytical or Monte Carlo projection integral. The spatial sound filter can then be efficiently computed using a convolution of the incoming sound field and the user's head-related transfer function (HRTF). This approach can compute spatial sound for area and volume sources $2-3$ orders of magnitude faster than a naïve approach.

Second, we present a technique that enables the HRTF to be applied to the sound propagation impulse response with low latency (Section 5.4). A perceptual metric is used to adaptively determine the number of spherical harmonic coefficients that are needed to represent the incoming sound field for each partition of the

[1]Much of this chapter appeared previously Schissler et al. (2016, 2017b)

impulse response. Then, we can efficiently convolve the HRTF with each partition for those SH coefficients to generate the spatial impulse response. By using fewer coefficients for less directional partitions, our technique saves significant computation and is more than an order of magnitude faster than the previous approach.

5.2 Background

In this section we present relevant background on spatial sound and sound rendering.

5.2.1 HRTF Spatial Sound

The head-related transfer function (HRTF) uses a linear FIR filter to map the sound arriving from a direction $\vec{x}$ to the sound received at the entrance of each ear canal of the listener. In Cartesian coordinates, the HRTF is a function of two parameters: direction $\vec{x}$, and either time t or frequency ω. The HRTF direction may also be specified in spherical coordinates by the tuple (θ, ϕ) where ϕ is the azimuth and θ is elevation. We denote the time-domain HRTF for the left and right ears as $H_L(\vec{x}, t)$ and $H_R(\vec{x}, t)$. The frequency-domain HRTF is denoted by $H_L(\vec{x}, \omega)$ and $H_R(\vec{x}, \omega)$. In the frequency domain, the HRTF filter can be stored using the real and imaginary components of the Fourier transform of the time-domain signal, or can be represented by the magnitude response and a frequency-independent inter-aural delay. In the second case, a causal minimum-phase filter can be constructed from the magnitude data using the min-phase approximation (Kulkarni et al., 1995) and the inter-aural delay. HRTFs are typically measured over evenly-spaced directions in anechoic chambers using specialized equipment. The output of this measurement process is an impulse response for each measured direction $\vec{x}_i$. We refer to this HRTF representation as a *sampled* HRTF. Another possible HRTF representation is one where the sampled HRTF data has been projected into the *spherical harmonic* orthonormal basis (Evans et al., 1998; Rafaely and Avni, 2010).

To compute spatial audio for a point sound source using the HRTF, we first determine the direction from the center of the listener's head to the sound source $\vec{x}_S$. Using this direction, the HRTF filters $H_L(\vec{x}_S, t)$ and $H_R(\vec{x}_S, t)$ for the left and right ears are interpolated from the nearest measured impulse responses. If the anechoic audio for the sound source is given by $s(t)$, and the sound source is at a distance r_S from the listener, the sound signals at the left ear $q_L(t)$ and the right ear $q_R(t)$ can be computed as follows:

$$q_L(t) = \frac{1}{1+r_S} H_L(\vec{x}_S, t) \bigotimes s(t) \tag{5.1}$$

$$q_R(t) = \frac{1}{1+r_S} H_R(\vec{x}_S, t) \bigotimes s(t) \tag{5.2}$$

where $\bigotimes$ is the convolution operator and $\frac{1}{1+r_S}$ is the distance attenuation factor. The $+1$ in the denominator is used to avoid a singularity near the sound source. Other distance attenuation models may also be used to suit the requirements of a specific application. If there are multiple sound sources, the signals for each source are added together to produce the final audio at each ear. For the sake of clarity, from this point forth, we drop the subscripts L and R of the HRTF. The reader should assume that the audio for each ear can be computed in the same way.

5.2.2 Spatial Impulse Response (SIR) Construction

When sound propagation is simulated within an environment, the output at each simulation step is called an *impulse response* (IR). The IR contains only the effect of the environment on the sound heard at the listener and can be represented in a few different ways. Wave-based sound propagation systems usually compute the IR as an array of time-domain pressure samples, $p(t)$. The monaural sound heard by the listener can be directly obtained by convolving $p(t)$ with the source's anechoic audio. To support directional listeners for wave-based sound propagation, the plane-wave decomposition of the pressure field can be used to spatialize the pressure impulse response (Mehra et al., 2014).

On the other hand, geometric sound propagation systems usually compute the IR in the sound intensity or sound energy domain for octave frequency bands, rather than directly in the pressure domain. In this case, the IR can be represented as a list of N_i *sound paths* which correspond to the reflection or diffraction paths detected on the current frame via ray tracing. The ith path contains the following information:

- $I_{i\bar{\omega}}$ - the sound intensity for the ith path and sound propagation simulation frequency band $\bar{\omega}$.
- t_i - the time of arrival or delay time for the path.
- $\vec{x}_i$ - the Cartesian 3D direction from the listener's position in the direction of sound arrival.

We use this representation in our spatial sound approach.

The canonical way to incorporate the HRTF into the spatial impulse response is to interpolate the HRTF filter for each sound path direction in the IR and then multiply by the pressure magnitude for the path (Kuttruff, 1993). We refer to this as the *per path* SIR construction method. The pressure SIR can be computed for frequency band $\bar{\omega}$ according to the relation,

$$p_{\bar{\omega}}(t) = \sum_{i=0}^{N_i} H(\vec{x}_i, t) \otimes \delta(t - t_i) \sqrt{I_{i\bar{\omega}} z_0} \tag{5.3}$$

where z_0 is the characteristic specific acoustic impedance of the propagation medium. This is essentially a direct time-domain convolution of the HRTF with the IR. To compute the final spatial pressure IR containing all frequency and direction-dependent sound propagation effects, the IRs for all simulation frequency bands must be band-pass filtered into their corresponding frequency bands and then summed:

$$p(t) = \sum_{\bar{\omega}} BandPass(p_{\bar{\omega}}(t), \bar{\omega}). \tag{5.4}$$

Alternatively, the HRTF can be filtered into separate frequency bands $H_{\bar{\omega}}(\vec{x}, t)$ in a preprocessing step to eliminate the need for filtering at runtime.

This generates an SIR that can be convolved with the anechoic source audio to produce the sound heard by the listener at its current position and orientation. A significant drawback of this method of SIR generation is that the HRTF must be interpolated for every sound path, and the number of paths can be more than 10^5. It can take over 500ms to compute the SIR for a single sound source in an optimized implementation (Lentz et al., 2007). As a result, this technique is not suitable for interactive applications. It is also possible to cluster paths based on their direction to reduce the number of interpolations (Lentz et al., 2007), but this reduces the quality and resolution of the spatial sound and is still too slow to meet the 100ms latency target for long impulse responses. An alternative approach that is commonly used in interactive auralization systems to save computation is to spatialize only the direct sound or early reflections with the HRTF, while the remainder of the IR uses amplitude panning. However, this results in late reverberation that is less spacious due to the lack of frequency-domain filtering and interaural time differences. It can also be difficult to closely match the timbre of the HRTF and panning parts of the IR.

5.2.3 Spherical Harmonics and Spatial Sound

The spherical harmonics (SH) are a set of orthogonal basis functions for the spherical domain and are denoted by $Y_{lm}(\vec{x})$, where $\vec{x}$ is a unit-length Cartesian direction, $l = 0, 1, ...n$ and $m = -l, ..., 0, ...l$. n represents the maximum spherical harmonic order. For order n, there are $(n+1)^2$ basis functions. An arbitrary spherical function $f(\vec{x})$ can be projected into the SH basis by evaluating an integral over the spherical domain to generate SH coefficients f_{lm}:

$$f_{lm} = \iint_{\mathbb{S}} Y_{lm}(\vec{x}) f(\vec{x}) d\mathbb{S}. \tag{5.5}$$

This integral can be evaluated using the discrete spherical harmonic transform or Monte Carlo numerical integration (Zotkin et al., 2009; Rafaely and Avni, 2010). With the Monte Carlo method, the SH coefficients are computed as a weighted sum of basis functions evaluated at a set of N uniformly-distributed random samples $\vec{x}_i$:

$$f_{lm} = \frac{1}{\sum_{i=0}^{N} f(\vec{x}_i)} \sum_{i=0}^{N} Y_{lm}(\vec{x}_i) f(\vec{x}_i). \tag{5.6}$$

Once the function is transformed into the SH basis, an approximation of the function, $\tilde{f}(\vec{x})$, can be computed in any direction $\vec{x}$:

$$f(\vec{x}) \approx \tilde{f}(\vec{x}) = \sum_{l=0}^{n} \sum_{m=-l}^{l} Y_{lm}(\vec{x}) f_{lm}. \tag{5.7}$$

If this process is applied to the head-related transfer function $H(\vec{x}, t)$, the result for each ear is a set of SH coefficients $h_{lm}(t)$. These coefficients can then be used to reconstruct an approximation of the HRTF:

$$H(\vec{x}, t) \approx \tilde{H}(\vec{x}, t) = \sum_{l=0}^{n} \sum_{m=-l}^{l} Y_{lm}(\vec{x}) h_{lm}(t) \tag{5.8}$$

Due to the orthogonality of the spherical harmonics, if the sound arriving at the listener at a time sample from all directions is expressed in SH coefficients X_{lm}, then the HRTF for that time sample can be efficiently computed using a dot product of the basis function coefficients:

$$\tilde{H}(t) = \sum_{l=0}^{n} \sum_{m=-l}^{l} X_{lm} h_{lm}(t) \tag{5.9}$$

This property of the spherical harmonics is important for the efficient application of the HRTF to area/volume sources and sound propagation impulse responses.

5.3 HRTF-based Spatial Sound for Area and Volume Sources

Although many current spatial audio systems support the use of HRTFs for point sources (Møller, 1992), few systems handle sound sources represented by an emissive area or volume in space. In these scenarios, the sound heard by the listener is a combination of sound from many directions, each with a different HRTF filter. For instance, an area sound source such as a river emits sound from the entire water surface. This gives the listener the impression that the source is extended in space along the direction of the river's flow, rather than being localized at a single point. In a forest, a listener might hear wind blow through the trees, creating a broad soundscape. Environments such as rivers or forests contain large area or volume sound sources that are difficult to recreate with traditional spatial audio techniques.

A naïve point-sampling approach might approximate a large source by a collection of many evenly-distributed point sources. For the island scene (Figure 5.8, top right), the ocean coastline sound source occupies an area of roughly 100,000m^2. Representing this sound source at a 1 meter resolution with points would require about 100,000 point sources and would take about 600 ms to compute. Users perceive this latency in head-tracked spatial audio in terms of source-lag and source-motion as the users's head rotates and this latency significantly detracts from users' experience in interactive VR applications (Brungart et al., 2004, 2005). On the other hand, a coarser point sampling would cause audible artifacts in the sound: When the listener is close to or inside the sound source, they would hear discrete sound coming from closest point sources rather than accurate directional audio coming from a region of space. A more sophisticated approach is needed to handle these challenging sources at the low latency required for head-tracked spatial audio in virtual reality applications.

5.3.1 Theoretical derivation

To simplify the discussion, we start with the following scenario illustrated in Figure 5.1: an area-volumetric sound source (S) and a listener (L). One method to compute spatial audio produced by this source at the listener is to sample the source with N_i discrete points. These point sources are at a distance $[r_1, r_2, ..., r_{N_i}]$ from the listener in the directions $[\vec{x}_1, \vec{x}_2, ..., \vec{x}_{N_i}]$. The spatial audio from the collection of

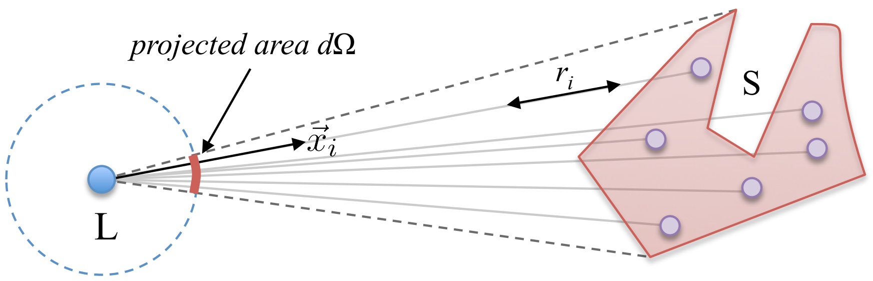


Figure 5.1: Visualization to illustrate the projection of area-volumetric sound source S on a sphere around the listener L. The source is approximated by many individual point sound sources, each described by direction $\vec{x}_i$ and distance r_i.

point sources can be computed as the summation of spatial audio produced by the individual point source (equation 5.1) to give:

$$p(t) = \frac{1}{N_i} \sum_{i=1}^{N_i} \left(\frac{1}{1+r_i} H(\vec{x}_i, t) \right) \bigotimes s_i(t), \tag{5.10}$$

where $H(\vec{x}, t)$ and $s_i(t)$ are the HRTF and the anechoic audio corresponding to the point source i, respectively. The factor $(1/N_i)$ is applied to normalize the amplitude of the area-volumetric sound source. Under the assumption that all point sources are emitting the same anechoic audio (i.e. $s_i(t) = s(t)$), the above equation becomes

$$\begin{aligned} p(t) &= \left(\frac{1}{N_i} \sum_{i=1}^{N_i} \frac{1}{1+r_i} H(\vec{x}_i, t) \right) \bigotimes s(t), & (5.11) \\ &= H_{\text{area}}(t) \bigotimes s(t), & (5.12) \end{aligned}$$

where $H_{\text{area}}(t) = \frac{1}{N_i} \sum_{i=1}^{N_i} \frac{1}{1+r_i} H(\vec{x}_i, t)$ is the spatial audio filter corresponding to the area-volumetric source. This equation shows that spatial audio filter for an extended source can be expressed as a weighted summation of the HRTFs of the constituent point sources. This is a discrete approximation and converges to the exact solution as $N_i \to \infty$. The continuum solution can be written as:

$$H_{\text{area}}(t) = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} X(\theta, \phi)\, H(\theta, \phi, t)\, \sin(\theta)\, d\theta\, d\phi, \tag{5.13}$$

where $X(\theta, \phi)$, also called the projection function, is a direction-dependent normalized weight function applied to the HRTF. It represents the distribution of sound pressure arriving at the listener position from all directions.

One way to think about the projection function is to visualize the projection of the area-volumetric source onto an imaginary sphere around the listener (Figure 5.1). The area-volumetric source will project onto an area of the listener's sphere (denoted by $d\Omega$). The projection function would have a non-zero value for all directions inside this projected area and have a zero value for directions outside. The projection value at any single direction inside the projection area $d\Omega$ depends on the sound radiation of the source and its distance attenuation in that direction. A significant advantage of considering the projection of the source, rather than doing a point-based sampling approach, is that the complexity of the approach is not based on the size of the sound source, only the projected area.

To compute the spatial audio filter for an area-volumetric sound source, we have to solve the integral equation (5.13). Solving this integral directly can be computationally expensive. The key insight of our work is to use orthonormal basis functions to solve this integral efficiently. The projection function $X(\theta, \phi)$ and HRTF $H(\theta, \phi, t)$ are both functions defined over a spherical domain (θ, ϕ). Similar to how a 1D signal can be expressed in terms of orthonormal Fourier bases, functions defined over a spherical domain can also be expressed in terms of orthonormal basis functions. We express the projection function $X(\theta, \phi)$ and HRTF $H(\theta, \phi, t)$ in orthonormal basis Ψ_{lm} as follows:

$$X(\theta,\phi) = \sum_{l=0}^{n} \sum_{m=-l}^{l} X_{lm} \;\; \Psi_{lm}(\theta,\phi) \tag{5.14}$$

$$H(\theta,\phi,t) = \sum_{l=0}^{n} \sum_{m=-l}^{l} h_{lm}(t) \;\; \Psi_{lm}(\theta,\phi) \tag{5.15}$$

Using the properties of orthonormal basis functions (see supplemental material), we can simplify the projection integral equation to give:

$$H_{\text{area}}(t) = \sum_{l=0}^{n} \sum_{m=-l}^{l} X_{lm} \;\; h_{lm}(t). \tag{5.16}$$

The same derivation holds for a frequency-domain HRTF representation $H(\theta, \phi, \omega)$ or an equivalent representation (such as minphase). Therefore, the spatial audio filter corresponding to the area-volumetric source can be computed as a dot product of the basis coefficients of the projection function and the listener's HRTF. To summarize, the steps required are as follows:

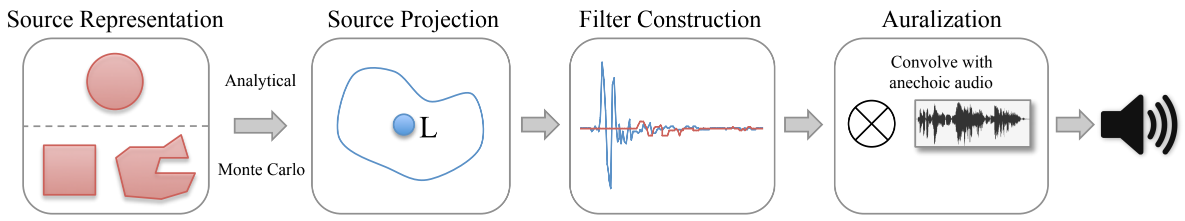


Figure 5.2: Overview of our spatial sound pipeline for area and volume sound sources. The pipeline is duplicated for each sound source in a scene. At runtime, the set of shapes for a source is first projected into the spherical harmonic basis using either the analytical formulation (spheres) or Monte Carlo ray sampling (boxes, meshes). This produces a set of basis coefficients that approximate the sound contribution from all shapes of the source. Next, HRTF filters are constructed for the left and right channels based on this projection. Finally, these filters are convolved with the anechoic input sound to produce the final audio for that sound source.

Note that the basis coefficients of the HRTF do not change at runtime and can be precomputed and stored. On the other hand, the basis coefficients of the projection function change with listener orientation, source-listener distance, source directivity, etc., and must be recomputed at runtime. The above equations can use any orthonormal basis functions defined for the spherical domain, such as spherical harmonics or spherical wavelets. We chose the spherical harmonics as the orthonormal basis functions in this work.

5.3.2 System Overview

Figure 5.2 shows an overview of our technique. We start with an area-volumetric sound source, represented as a collection of geometric shapes. During the preprocessing step, the spherical harmonic (SH) coefficients of the HRTF are precomputed and stored for runtime use. At runtime, given a set of input shapes that constitute an area-volumetric source, we determine the projection of each shape on the spherical domain centered at the listener. Next, the projection coefficients computed for each shape individually are then summed up for all the shapes constituting the source. The spatial audio filter for this area-volumetric source is computed as a dot product of the SH coefficients of the projection function and the HRTF. The spatial audio filter for each sound source is then convolved with the input anechoic audio of the source to generate spatial audio for the source. This pipeline is duplicated for each area-volumetric sound source in the scene and the spatial audio for all the sources are summed together to generate the final sound to be played over the headphones. We now discuss each step in detail.

Source Representation: In our technique, an area-volumetric source is defined as a collection of one or more geometric shapes that emit sound from an area or volume. During the scene design phase, an artist or a game designer could either (a) place these geometric shapes in the scene and create an area-volumetric sound source as their collection or (b) select part of the scene geometry (river, forest) and assign it as an area-volumetric sound source. The geometric shapes associated with an area-volumetric source are: (a) sphere, (c) box, and (c) arbitrary mesh. Shapes (a) and (b) are volumetric sources, whereas (c) could be an area (open mesh) or volumetric source (closed mesh). The union of multiple shapes can describe complex sound sources (see Figure 5.3).

For an area source, sound is emitted uniformly from all surfaces with distance attenuation based on the distance to the surface. If a sound source is a closed volume (e.g. sphere, box, arbitrary mesh), the sound is emitted uniformly within the volume, with distance attenuation outside the volume. Each area-volumetric source has a spatial audio filter (that need to be computed) and a stream of anechoic unprocessed audio samples. At runtime, each source results in one convolution operation between its spatial audio filter and the anechoic audio.

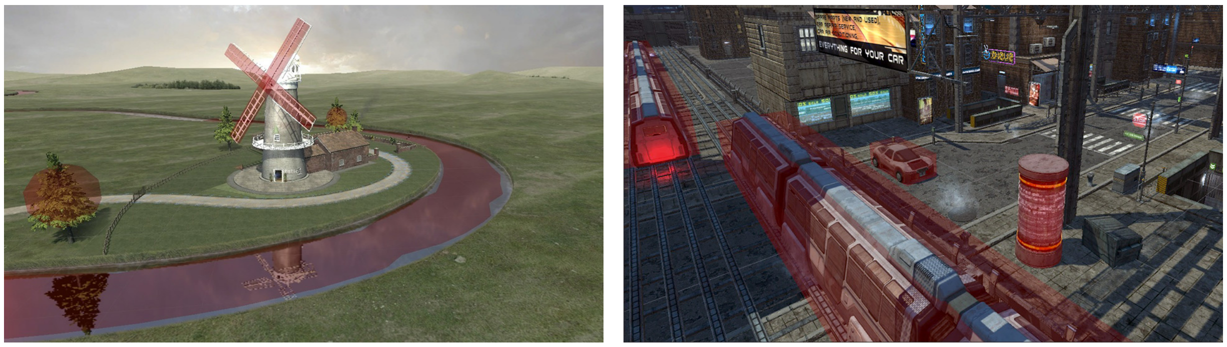


Figure 5.3: This visualization shows the sound sources for the windmill and city scenes in red. In the windmill scene, box sound shapes are used to represent the windmill sails, spheres are used for trees, and a triangle mesh is used for the nearby river. In the city, the train and car sound sources are represented by boxes, while scrolling advertisements are represented using meshes that correspond to the visual geometry.

Source Projection: In this step, each source shape is projected into a spherical domain centered at the listener and the the spherical harmonic coefficients of the projection function are computed. This involves solving the following integral equation for basis function coefficients X_{lm}:

$$X_{lm} = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi} f(\theta, \phi) Y_{lm}(\theta, \phi) \sin(\theta) \, d\theta d\phi. \tag{5.17}$$

In the special case of a spherical sound source, these coefficients can be computed analytically as a function of the radius of the sphere and its distance from the listener. Section 5.3.3 describes this analytical projection technique in detail.

In the case of a box or mesh, no such closed form solution exists and we have to compute the projection coefficients numerically. This is computed by using an efficient Monte-Carlo integration approach (Section 5.3.4). The number of rays used in this approach is determined adaptively based on the size of the projection area of the area-volumetric source shape. In other words, a source shape at a greater distance has a smaller projection area, and thus fewer rays are traced compared to a nearby source shape.

Filter Construction: The spatial audio filter construction process computes the dot product of the SH coefficients of the projection function of the source shape (determined in the previous step) with the SH coefficients of the HRTF. This step is repeated for each shape of sound source. The results sum to generate the filter for the corresponding area-volumetric source. This step is repeated for each ear to generate the spatial audio filter for the left and right ears.

Auralization: In this last step, the filters of the area-volumetric source are convolved with the anechoic audio associated with the source to generate binaural audio corresponding to that source.

5.3.3 Analytical Projection

In the special case of spherical source projection, the spherical harmonic coefficients of the projection function can be computed analytically. Let's take a scenario in which we have a spherical area-volumetric source of radius R at a distance r from the listener (Figure 5.4). We choose the listener's coordinate frame such that it is centered at the listener with the z axis oriented toward the listener-source direction.

Note that a sphere's projection over another sphere is a circular projection area. The radius of this projection area is independent of the orientation of the source sphere and depends only on the radius R and distance r. Using trigonometry, we can relate R and r to the the half-angle of the projection, $\alpha = \sin^{-1}(R/r)$. Mathematically, the projection function $X(\theta, \phi)$ has the following form:

$$X(\theta, \phi) = X(\theta) \begin{cases} \neq 0 & : 0 \leq \theta < \alpha \\ 0 & : \text{otherwise} \end{cases}$$

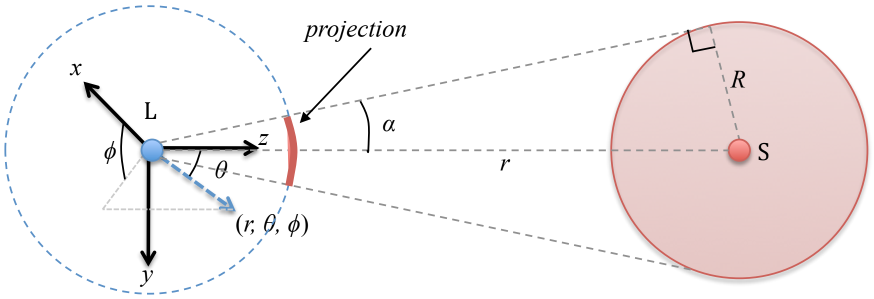


Figure 5.4: The geometry of the analytical source projection for listener L and spherical source S. The projection depends solely on the angle $\alpha = \sin^{-1}\left(\frac{R}{r}\right)$, where r is the distance to the sphere's center, and R is the sphere's radius. We choose a coordinate system for the projection with the z axis oriented in the direction of the sphere in order to simplify the derivation.

In other words, the projection function is non-zero inside the projection area and zero outside. The expression to evaluate the spherical harmonic coefficients of the projection function becomes:

$$X_{lm} = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\alpha} X(\theta)\, Y_{lm}(\theta, \phi)\ \sin\theta\ d\theta d\phi. \tag{5.18}$$

Since the definition of spherical harmonics changes with the order m, we have three cases:

Case: $m > 0$

Using the definition of spherical harmonics, we get

$$\begin{aligned} X_{lm} &= \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\alpha} X(\theta)\, \Gamma_{l|m|}\ P_l^{|m|}(\cos\theta)\ \cos(|m|\phi)\ \sin\theta\ d\theta d\phi \\ &= \Gamma_{l|m|} \int_{\theta=0}^{\alpha} X(\theta)\ P_l^{|m|}(\cos\theta)\ \sin\theta\ d\theta \quad \int_{\phi=0}^{2\pi} \cos(|m|\phi)\ d\phi \end{aligned}$$

The right side expression $\int_{\phi=0}^{2\pi} \cos(|m|\phi)\ d\phi = \left[\frac{\sin(|m|\phi)}{m}\right]_0^{2\pi} = 0$. Therefore, $X_{lm} = 0$ for $m > 0$.

Case: $m < 0$

Using similar derivation as above, it can be shown that $X_{lm} = 0$ for $m < 0$.

Case: $m = 0$

The SH coefficients for the case $m = 0$ are referred to as the zonal harmonics coefficients z_l.

$$X_{l0} = z_l = \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\alpha} X(\theta)\, \Gamma_{l0}\, P_l^0\left(\cos\theta\right)\ \sin\theta\ d\theta d\phi$$

Substituting $z = \cos\theta$ gives us:

$$z_l = \int_{\phi=0}^{2\pi} \int_{z=\cos\alpha}^{1} X(\cos^{-1} z)\, \Gamma_{l0}\, P_l^0\left(z\right)\ dz\ d\phi$$

Separating variables:

$$z_l = \Gamma_{l0} \int_{z=\cos\alpha}^{1} X(\cos^{-1} z)\, P_l^0\left(z\right)\ dz\quad \int_{\phi=0}^{2\pi} d\phi$$

Integrate by parts:

$$z_l = 2\pi\Gamma_{l0}\left[X(\cos^{-1} z)\, Q_l^0(z) - \int \frac{d}{dz}(X(\cos^{-1} z))\, Q_l^0(z)\, dz\right]_{\cos\alpha}^{1} \tag{5.19}$$

where

$$Q_l^0(z) = \int P_l^0\left(z\right)\ dz = 2\sum_{k=0}^{l} \beta_{lk}\, B_{\frac{z+1}{2}}(1, k+1).$$

The notation $B_z(a,b) = \frac{1-(1-z)^b}{b}$ is the incomplete beta function and $\beta_{lk} = \frac{(-l)_k\ (l+1)_k}{k!\ k!}$ is a constant where $(x)_n$ is the Pochhammer symbol.

The right hand side expression in equation 5.19 above can be analytically integrated if the term $\frac{d}{dz}(X(\cos^{-1} z))$ is a constant. This implies that we can compute SH coefficients of the projection function analytically if the projection function is of the form $X(\cos^{-1} z) = cz + d$ where c and d are constants[2].

In case of spherical sources, a projection function with maxima at $\theta = 0$ and minima for $\theta = \alpha$ would ensure that the projection value is proportional to the depth of the source in that direction. For this purpose, we choose a function such that $\frac{d}{dz}(X(\cos^{-1} z)) = \frac{1}{1+r}\ \frac{1}{1-\cos\alpha}$, such that $X(\theta) = \frac{1}{1+r}\ \frac{\cos\theta - \cos\alpha}{1-\cos\alpha}$. Using this projection function in equation 5.19, simplifies it to:

$$z_l = \frac{1}{1+d}\ \frac{4\pi}{1-\cos\alpha}\ \sum_{k=0}^{l} \beta_{lk}\left[\frac{1-\cos\alpha}{k+1} - D(1,k) + D(\cos\alpha, k)\right], \tag{5.20}$$

[2]This logic can be applied recursively to support a function whose m^{th} order derivative is constant i.e. $\frac{d^m}{dz^m}(X(\cos^{-1} z))$ is constant. This would give projection functions of the form $X(\cos^{-1} z) = a_m z^m + a_{m-1} z^{m-1} + ... + a_0$.

where

$$D(a,b) = \frac{2^{-b-1}(1-a)^{b+2}}{(b+1)(b+2)} + \frac{a}{b+1}. \tag{5.21}$$

Using the above derivation, the SH coefficients X_{lm} of the projection function $X(\theta,\phi)$ are defined as follows:

$$X_{lm} = \begin{cases} 0 & : m > 0 \\ z_l & : m = 0 \\ 0 & : m < 0 \end{cases} \tag{5.22}$$

When the listener's head is at a different orientation $(\theta_{list}, \phi_{list})$ with respect to the orientation assumed in Figure 5.4, the spherical harmonic coefficients of the projection function can be computed by using the zonal harmonics rotation equation as follows:

$$X_{lm} = \sqrt{\frac{4\pi}{2l+1}}\; z_l \;\; Y_{lm}(\theta_{list}, \phi_{list}). \tag{5.23}$$

The outcome is a set of analytically computed SH coefficients X_{lm} of the projection function that can be used to construct the spatial audio filter for a spherical sound source.

The above case only works when the listener is outside the spherical source. The case when the listener is inside must be handled separately because the value of $\alpha = \sin^{-1}\left(\frac{R}{d}\right)$ is undefined. Inside the source, sound arrives at the listener from all directions and the directivity of the source is reduced. There is a smooth transition in the spatial audio as a listener moves from the edge of the sphere toward the center, with more directivity at the edge and less near the center. To acheive this effect inside the source, we first compute the analytical SH projection coefficients X_{lm} as if the listener was at the closest point on the sphere's surface where $\alpha = \frac{\pi}{2}$. This produces coefficients with strong directivity. Then we attenuate the resulting coefficients X_{lm} by the factor $\frac{d}{R}$ for $l > 0$, leaving the DC coefficient X_{00} with constant directivity unchanged. Toward the center, $X_{lm} \to 0$ for $l > 0$. As a result the directionality of the sound source reduces naturally as the listener approaches the sphere center.

5.3.4 Monte Carlo Projection

While spheres are rotationally invariant and allow for an analytical projection formulation, this is not true for more complex sound sources. In this section, we describe an efficient Monte Carlo based formulation to solve the projection integral, Equation 5.17, for arbitrary shapes.

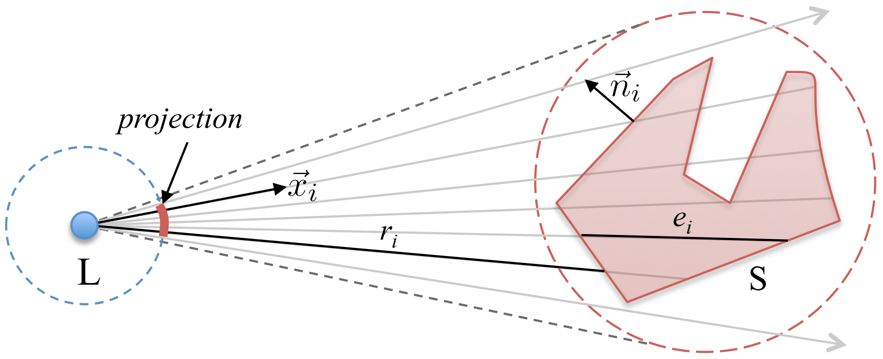


Figure 5.5: The Monte Carlo projection uses rays to sample the sound contribution from arbitrarily-shaped sources. When the listener L is outside the bounding sphere of the sound source, S, we trace rays in the cone defined by the source's bounding sphere. Inside the bounding sphere, rays are traced in all directions uniformly. Each ray is given a weight w_i that is used to estimate the value of the projection integral. If a ray does not hit the source, that ray has $w_i = 0$.

Background: In Monte Carlo integration, a set of uniformly distributed random samples are used to numerically compute the integral of a function. Each sample is weighted according to its probability. An approximate value for the integral is computed by summing the weighted random samples. Due to the law of large numbers, the accuracy of the integral increases when more samples are taken. This approach has previously been applied for computing direct light for computer graphics (Shirley and Wang, 1994), as well as for low-order spherical harmonic representations of lighting (Green, 2003). In our approach, we modify this formulation to efficiently compute the projection of an area-volumetric sound source.

Monte Carlo Projection for Arbitrary Shapes: We present a Monte Carlo numerical integration technique that computes an approximation of the SH coefficients of a source's projection function using a set of random rays. This operation is performed for each of a sound source's shapes independently and the results are added to produce the SH coefficients for the entire source. Our approach begins by generating a set of N_i uniformly-distributed rays with directions $\vec{x}_i = (\theta_i, \phi_i)$ that sample the bounding sphere of a complex area or volumetric sound source shape. This process is illustrated in Figure 5.5. The rays are intersected with the geometry of the source and used to compute the projection of the source at the listener's spherical domain. Each ray is weighted by a factor w_i that specifies how much that ray contributes to the final projection. For area sound sources (e.g. triangle meshes), $w_i = X(\theta_i, \phi_i)$ is proportional to the distance attenuation from the

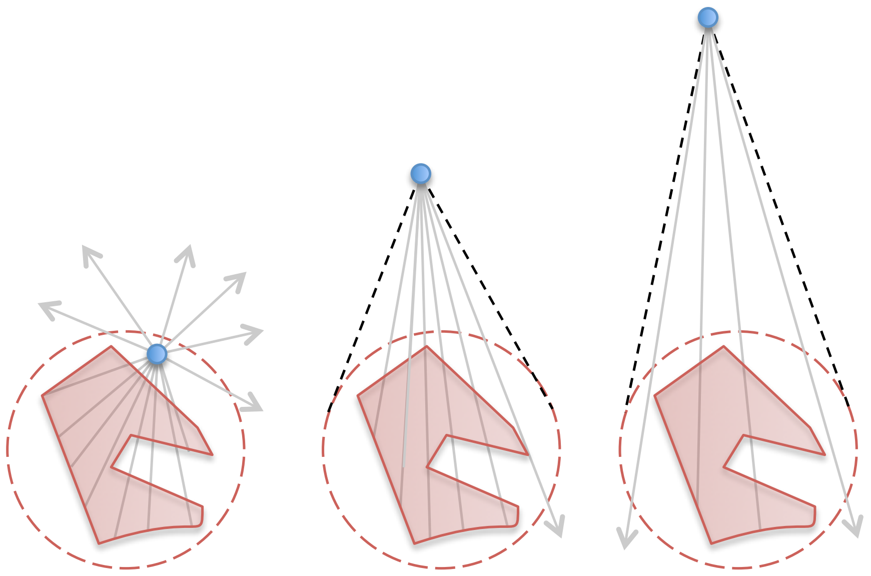


Figure 5.6: The number of rays used to computed the Monte Carlo projection changes depending on the distance from the listener to the sound source's bounding sphere. If the listener is inside the bounding sphere (left), many rays are traced in all directions. Outside the bounding sphere, the number of rays that are used for Monte Carlo integration decreases proportional to the solid angle of the bounding sphere.

ray's intersection point to the listener, $\frac{1}{1+r_i}$, as well as the dot product of the ray direction $\vec{x}_i$ with the surface normal vector $\vec{n}_i$.

$$w_i = \left(\frac{1}{1+r_i}\right) \max(-\vec{x}_i \cdot \vec{n}_i,\ 0) \qquad \text{(area sources)} \tag{5.24}$$

For volumetric sound sources, we choose w_i to also include the distance the ray travels through the source, e_i:

$$w_i = e_i \left(\frac{1}{1+r_i}\right) \max(-\vec{x}_i \cdot \vec{n}_i, 0) \qquad \text{(volume sources)} \tag{5.25}$$

If a ray does not intersect a sound source or is blocked by an obstacle in the scene, we set $w_i = 0$ for that ray. The spherical harmonic coefficients X_{lm} of the projection can then be computed by the following equation:

$$X_{lm} = \frac{1}{\sum_{i=0}^{N_i} w_i} \sum_{i=0}^{N_i} w_i\, Y_{lm}(\vec{x}_i). \tag{5.26}$$

As an optimization, we trace fewer rays for distant sound sources, as shown in Figure 5.6. The number of rays, N_i, is chosen to be proportional to the solid angle of the source's bounding sphere. This saves computation for distant sound sources while maintaining the same sampling density from the listener's point of view.

When the listener is inside the bounding sphere of the shape, the source is sampled using uniform random rays in all directions with the same sampling density.

5.4 Efficient Perceptual Construction of the Spatial Impulse Response

The efficient computation of the spatial impulse response (SIR) is a challenging problem when generating sound for interactive virtual reality. The SIR must be updated at a rate that is fast enough for the user to notice no perceptible latency. A commonly used threshold for the maximum end-to-end system latency is roughly 100ms (Sandvad, 1996; Lindau, 2009). This means that the total time it takes to recompute the sound propagation IR, apply spatial sound to generate a SIR, interpolate the convolution system to the new SIR, and reproduce the audio through headphones must be less than the maximum latency. If not, sound can seem to lag behind the user's current head position. While much work has been done to reduce the latency of sound propagation for interactive applications, previous techniques for generation of the SIR from the sound propagation IR may take over 500ms or more for a single sound source and therefore are too slow to meet this latency target (Lentz et al., 2007).

To overcome this obstacle, we present a novel technique for computation of the SIR that is about an order of magnitude faster than previous approaches. In Figure 5.7, we summarize our algorithm. The input to our approach is an impulse response (IR) that has previously been computed using a geometric sound propagation system. The sound paths in the IR are sorted into partitions of length L, and for each partition we evaluate the directivity strength using a perceptual metric based on the user's threshold of hearing and a spherical harmonic representation of the HRTF. Our metric adaptively determines the minimum spherical harmonic order $\tilde{n}$ that is required to represent the partition's spatial sound with no perceptible loss in quality. Then, we efficiently convolve the HRTF and the IR partition in the spherical harmonic domain up to order $\tilde{n}$ to generate the SIR for the partition. Finally, this partition SIR is overlap-added at the corresponding position in the output SIR. When all partitions have been processed, the result is a spatial pressure impulse response that can be convolved with anechoic source audio to render the sound at the listener's position.

5.4.1 Perceptual Directivity Metric

A main component of our approach is a novel metric that evaluates the minimum required spherical harmonic order $\tilde{n}$ for each partition in the room impulse response. Our metric works by examining the

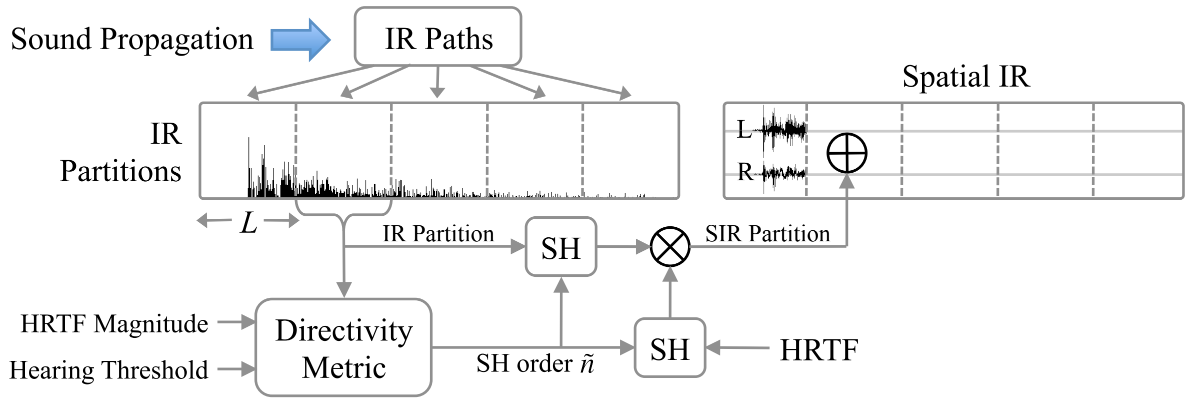


Figure 5.7: A visual representation of our spatial room impulse response construction algorithm. The IR output of sound propagation is split into partitions with size L, and the directivity of each partition is evaluated to determine the minimum spherical harmonic (SH) order $\tilde{n}$ for the partition's spatial sound. The partition's IR is converted to the SH basis up to order $\tilde{n}$, and then convolved with a SH representation of the HRTF up to order $\tilde{n}$. The resulting filters for the left and right channels are added to the output SIR at the partition's offset in the IR.

spatial distribution of sound energy arriving at the listener during the partition. If there is strong directional information in the partition, then a higher SH order will be required to accurately represent the sound field. Otherwise, if the partition is more diffuse, a lower SH order can be used that requires less computation. In most indoor environments, earlier partitions will tend to be more directional, while the later ones will be more diffuse. Our metric takes advantage of this property of the IR so that the expensive high-order HRTF is used only where necessary. A key feature of our metric is that it can be evaluated very efficiently, so that the time saved by using a low-order HRTF for some partitions outweighs the time spent evaluating the metric.

Given N_i sound paths that arrived during a partition, the metric first computes the distribution of sound energy incident at the listener's position for each of the simulation frequency bands. The result is $X_{lm\bar{\omega}}$, a set of normalized SH coefficients for each simulation frequency band $\bar{\omega}$ up to a maximum SH order n_{max}. $X_{lm\bar{\omega}}$ can be computed using a form of Monte Carlo integration:

$$X_{lm\bar{\omega}} = \frac{1}{\sum_{i=0}^{N_i} I_{i\bar{\omega}}} \sum_{i=0}^{N_i} Y_{lm}(\vec{x}_i) I_{i\bar{\omega}}. \tag{5.27}$$

Here, the basis functions are evaluated for each path's direction and then weighted by the path's intensity at each frequency band.

Next, we use this energy distribution and the user's HRTF to determine a magnitude response of the SIR partition at each frequency band. As a preprocessing step, the frequency-domain HRTF $H(\vec{x}, \omega)$ is transformed into the SH basis to generate coefficients $h_{lm}(\omega)$. Then, the average magnitude response over each simulation frequency band $\bar{\omega}$ is computed to yield a spherical harmonic representation of the HRTF's magnitude response for that band, $h_{lm\bar{\omega}}$. Using the orthogonality property of the spherical harmonics (5.9), an approximation of the magnitude of the SIR partition can be computed:

$$\left|\tilde{H}_{\bar{\omega},n}\right| = \sum_{l=0}^{n} \sum_{m=-l}^{l} X_{lm\bar{\omega}} h_{lm\bar{\omega}}, \tag{5.28}$$

where $\left|\tilde{H}_{\bar{\omega},n}\right|$ is the pressure magnitude of the sound arriving during the partition for frequency band $\bar{\omega}$ and SH order n. This relationship is used to efficiently evaluate the impact of using a given spherical harmonic order n on the resulting spatial sound. The goal of the metric is to determine SH order $\tilde{n} \leq n_{max}$ such that $\left|\tilde{H}_{\bar{\omega},\tilde{n}}\right|$ is perceptually indistinguishable from $\left|\tilde{H}_{\bar{\omega},n_{max}}\right|$. More precisely, the metric must satisfy the condition $\left|\left|\tilde{H}_{\bar{\omega},\tilde{n}}\right| - \left|\tilde{H}_{b,n_{max}}\right|\right| < \epsilon$ where ϵ is a perceptually-based threshold.

One possibility is to compare against the absolute human threshold of hearing. The threshold is an important psychoacoustic quantity that corresponds to the smallest sound pressure level that a human can perceive at a given frequency (Fletcher, 1940; Robinson and Dadson, 1957). The threshold for the average adult listener, $T_q(\omega)$, can be analytically approximated as a function of frequency using the following relation (Terhardt, 1979):

$$\begin{aligned} T_q(\omega) =& 3.64(\omega/1000)^{-0.8} - 6.5e^{-0.6(\omega/1000-3.3)^2)} + \\ & 10^{-3}(\omega/1000)^4. \text{ (db SPL)} \end{aligned} \tag{5.29}$$

We use this function to determine the maximum allowed error in the spatial sound for a given frequency band in units of pascals. Alternatively, the user's threshold of hearing can be measured using standard audiometric techniques and then interpolated to get the threshold at an arbitrary frequency. A graph of the average hearing threshold for an adult listener is shown in Figure 3.6. The final relationship that must be satisfied is then:

$$|p_{\bar{\omega}}| \left|\left|\tilde{H}_{\bar{\omega},\tilde{n}}\right| - \left|\tilde{H}_{\bar{\omega},n_{max}}\right|\right| < T_q(\bar{\omega}), \tag{5.30}$$

where $|p_{\bar{\omega}}| = \sqrt{z_0 \sum_{i=0}^{N_i} I_{i\bar{\omega}}}$ is the total pressure magnitude for the partition. To determine the value of $\tilde{n}$ using the threshold of hearing, the metric starts at SH order $\tilde{n} = 1$, and then evaluates equation 5.30 for successively higher orders until the threshold is satisfied. The result is SH order $\tilde{n}$ that can be used to compute the SIR for the current partition. If $\tilde{n} < n_{max}$, then significant computation can be saved.

5.4.2 Convolution with the HRTF

Once the minimum spherical harmonic order $\tilde{n}$ has been determined for a given partition, the next step is to convolve the partition IR with the user's HRTF to generate the SIR for the partition.

First, the time-domain spherical harmonic signal for the IR partition must be computed from the N_i sound paths that arrived during the partition. This can be done by evaluating equation 5.27 for each time sample in the partition with the appropriate path delays added:

$$X_{lm\bar{\omega}}(t) = \frac{1}{\sum_{i=0}^{N_i} \delta(t - t_i) I_{i\bar{\omega}}} \sum_{i=0}^{N_i} \delta(t - t_i) Y_{lm}(\vec{x}_i) I_{i\bar{\omega}}. \tag{5.31}$$

The result of this operation is a set of normalized SH coefficients for each time sample and frequency band in the partition that represent an approximation of the directional information up to SH order $\tilde{n}$.

Next, the energy-time curve for the partition, $I_{\bar{\omega}}(t)$, is computed as a sum of delayed impulses:

$$I_{\bar{\omega}}(t) = \sum_{j=0}^{M} \delta(t - t_i) I_{i\bar{\omega}}. \tag{5.32}$$

This signal represents the sound energy decay for the partition at frequency band $\bar{\omega}$.

To efficiently perform the convolution with the HRTF in frequency domain, the signals $X_{lm\bar{\omega}}(t)$ and $I_{\bar{\omega}}(t)$ which are of length L must be padded at the end with zeros so that they are $2L$ audio samples long. In a preprocessing step, the HRTF is padded with zeros in time domain so that it is also $2L$ samples long. The HRTF is converted to frequency domain with a forward Fourier transform of size $2L$ and then projected into the spherical harmonic basis, yielding complex HRTF coefficients $h_{lm}(\omega)$. The partition SIR for frequency band $\bar{\omega}$ can then be computed by convolving $h_{lm}(\omega)$ with the Fourier transform of the IR signals:

$$p_{\bar{\omega}}(t) = \mathcal{F}^{-1} \left[\sum_{l=0}^{\tilde{n}} \sum_{m=-l}^{l} h_{lm}(\omega) \mathcal{F} \left(X_{lm\bar{\omega}}(t) \sqrt{I_{\bar{\omega}}(t) z_0} \right) \right] \tag{5.33}$$

where $\mathcal{F}$ is the Fourier transform operator. The resulting filters for all frequency bands are then band-pass filtered and then summed according to equation 5.4 to generate the full SIR for the partition. Then, the partition SIR is added to the output SIR at the partition's time offset. When all partitions have been processed, the SIR is complete and can be convolved with the anechoic audio for the sound source.

5.5 Implementation

In this section we describe the various implementation details for our spatial sound rendering system. We implemented our approach as a plugin for the Unity™game engine. The sound sources, listener, and scene geometry are specified by attaching scripts to objects within the game engine. Spatial audio processing is applied to each sound source's anechoic audio as a custom Unity audio effect. The sound for all sources is then mixed for stereo reproduction using Unity's built-in audio mixing system. Our system supports dynamic area/volume sources, listeners, and moving rigid geometry.

Sound Propagation: The propagation of sound within the virtual scene is computed in 4 logarithmically-distributed frequency bands: $0 - 110$Hz, $110 - 630$Hz, $630 - 3500$Hz, and $3500 - 22050$Hz. We use the ray-based image source method (Vorländer, 1989) to compute early specular reflections and Monte Carlo path tracing from the listener for diffuse reflections. We accelerate these computations using temporal coherence techniques such as the specular path cache and the impulse response cache (Chapter 3). Our sound propagation module also computes diffraction up to order 3 according to the UTD model (Tsingos et al., 2001; Schissler et al., 2014) during the specular ray tracing step. The number of primary rays traced on each frame from the listener is calculated based on the time taken to compute the previous frame. This allows our system to adaptively reduce or increase the simulation quality to maintain a specific update time for sound propagation. About $500 - 1,000$ primary rays are traced for indoor scenes, while more rays are traced outdoors because most rays escape the scene after a few bounces. The ray tracing is parallelized across half of the available CPU threads (6 in this case), and these threads execute with low priority to avoid audio rendering glitches.

Area/Volume Sound Sources: We incorporate a simple model of sound propagation delay into our technique for direct sound of area and volume sources. Rather than using the actual delay to each audible point on sound source, we use the *minimum* delay to avoid comb-filtering artifacts which occur when the same anechoic source audio is played with slightly different delays. We compute the nearest sample on the source and use the delay to that point for sound rendering. This delay is then used in a fractional delay interpolation algorithm (Wenzel et al., 2000) to produce smoothly-varying sound that incorporates Doppler shifting effects for the direct sound. The anechoic sound for each source is resampled using linear interpolation and then convolved with the spatial audio filter. Other propagation delay models such as center of bounding sphere or center of gravity of mesh are also possible options.

Spatial IR Construction: In our implementation, we use an impulse response partition size of $L = 512$ samples, or roughly 10.7ms at a $48,000$kHz sampling rate. The filtering of the spatial IR into frequency bands is accomplished using a 4th-order time-domain Linkwitz-Riley crossover network. The SRIR(s) for all sources are computed in parallel using the other half of the CPU threads (6 in our system). Spatial IR construction is performed in parallel with sound propagation in order to reduce the update period, and the IR(s) are double-buffered such that the sound propagation for frame n is computed while the spatial IR for frame $n - 1$ is constructed.

Spherical Harmonics: Efficient computation of our spatial sound techniques requires fast evaluation of the real spherical harmonics. We use the formulation proposed in (Sloan, 2013) that uses aggressive constant propagation and recurrence relations to speed up the computation for normalized cartesian vectors. It is more than an order of magnitude faster than naïve SH evaluation and substantially reduces the computational requirements for our spatial sound techniques. In a recent study of HRTF localization, the 4th-order spherical harmonics were sufficient to achieve accurate localization performance (Romigh et al., 2015). As a result, we use a maximum spherical harmonic order of $n_{max} = 4$.

Sharp Directivities: Low-order spherical harmonics may not always be sufficient to represent cases where the impulse response has very sharp directivities, such as with direct and early reflected sound. To handle this, we implemented a simple approach that finds important propagation paths in a preliminary pass over the RIR and then performs accurate HRTF interpolation (5.3) for just those paths. The other paths are computed using the our approach from Section 5.4. A path is considered important if its intensity is a significant fraction of the total energy in the impulse response, e.g. 1%. This approach tends to reduce the SH order required to represent strongly directional impulse responses. As a result, a smaller value for n_{max} can be used to save time in evaluation of the directivity metric. However, we did not notice any significant impact on performance or sound quality in the benchmark scenes so this module was disabled for our main results.

Auralization: Given a set of spatial impulse responses, the auralization module uses a non-uniform partitioned block convolution algorithm to efficiently convolve the SRIRs with the anechoic audio for each sound source with low latency (Gardner, 1994). We use an initial block size of 64 samples, and then double the block size every 4 blocks until a maximum block size of 512 samples is reached. This keeps both the convolution latency (128 samples) and the latency to update the impulse response ($\leq$ 512 samples) low. A thread pool with 2 high-priority threads is used to execute the convolution for each group of 4 blocks in parallel, and the priority for each of the tasks is inversely proportional to the block size. On each audio rendering frame, the audio device output thread waits on the thread pool tasks that are due on that frame (Battenberg and Avizienis, 2011). When the IR for a block is updated, a convolution is computed for both the previous and next filters, and then the results are interpolated in time domain over the block length (Müller-Tomfelde, 2001). The resulting audio for all sound sources is mixed and then sent to the audio device for playback. We use the Unity™game engine audio system which introduces an additional 21.3ms of latency due to audio output buffering.

5.6 Results

In this section we present results and analysis of our spatial sound techniques. First we demonstrate the performance of our technique for area and volume sources. Then, we show how our method for efficient spatial IR construction can improve on existing approaches.

5.6.1 Area and Volume Sources

We evaluated the performance of our technique on four scenes with varying source complexity. The scenes are depicted in Figure 5.8, and the number and types of sources in each scene are shown in Table 5.1. The performance results are summarized in Table 5.2. The timings were measured on a single thread of a 3.4 GHz Intel Core i7-4930K CPU and were averaged over 1000 frames. For all scenes our method can update the spatial audio filters in less than 1 ms. We break down the total time into the time spent on the analytical source projection (spheres only) and Monte Carlo projection (boxes, meshes) for each scene. The source projection time scales linearly with the number of shapes for which the projection must be computed. On the

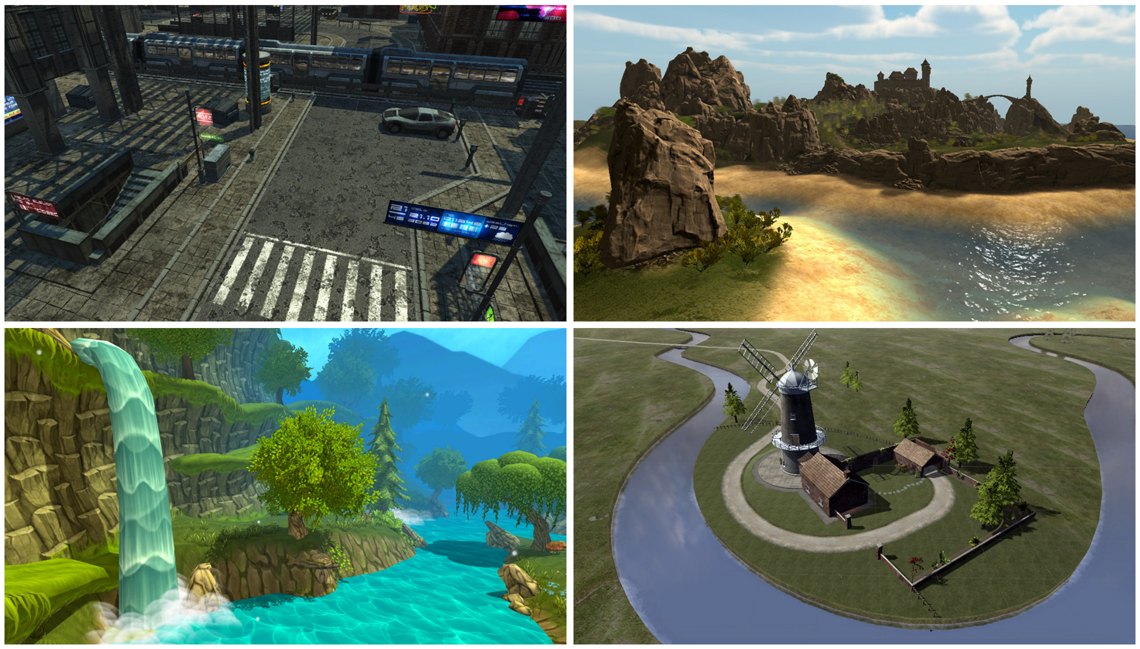


Figure 5.8: The four benchmark scenes used to evaluate our spatial sound technique for area and volume sources. The scenes contain various large sound sources such as trains, a car, an ocean, waterfalls, rivers, lakes, and windmills.

		Scene Complexity	
Scene	# Sources	# Shapes	Src area (m^2)
City	7	B(3), M(4)	0.1K + 1.6K
Windmill	4	S(2), B(4), M(1)	8K + 1K
Waterfalls	4	S(10), M(2)	0 + 30K
Island	5	S(43)	100K + 0

Table 5.1: The complexity of the scenes used to evaluate our area/volume spatial sound technique. The number of sound source shapes in each scene is specified using the notation: S=spheres, B=boxes, M=meshes. We list the estimated volume and area of all sound sources in each scene.

other hand, the filter construction is done only once per source. The memory usage of our technique is small. The primary cost is the HRTF storage, which uses 100KB when stored in the SH domain up to 9th order.

In Table 5.2, we also compare the performance of our method to a naïve point-source approximation (Equation 5.11). Using the area and volume of the sound sources given in Table 5.2, we estimate the computation time of this technique for each scene. Computing an HRTF filter for a single point source takes about 0.006 ms. If the sound sources are sampled using points at a coarse 1 meter resolution, our method outperforms point sources for all scenes. The difference is most noticeable for large sound sources in the Waterfall and Island scene. These scenes would require greater than 100 ms to compute using the

Scene	**Render load** (% CPU)	**Our technique (ms)** Analy. Proj.	M.C. Proj.	Filter const.	Total	**Point sampling (ms)** Total	**Speedup** (Naïve/Our)
City	3.30	-	0.09	0.09	**0.18**	**10**	**55**
Windmill	1.64	0.01	0.11	0.06	**0.18**	**54**	**300**
Waterfalls	1.57	0.13	0.13	0.05	**0.31**	**180**	**581**
Island	2.04	0.55	-	0.07	**0.62**	**600**	**968**

Table 5.2: The performance results of our area/volume spatial sound system. We report the computational load of the audio rendering thread (Render load) performing the auralization step. We report the timings for both the analytical projection (Analy. Proj.) used for spherical sources and Monte Carlo (M.C. Proj) used for box and meshes along with the filter construction cost (Filter const.). For all scenes, our approach can compute spatial sound filters in less than 1 millisecond. We also list the approximate time needed for the naive point-sampling approach. Point sources were sampled at a 1 meter resolution (filter computation time per point source = 0.006 ms). Our spatial sound algorithm is 2-3 orders of magnitude faster than the naïve approach.

point-sampling approach and would result in perceivable latency for VR applications (Brungart et al., 2005). Our approach takes less than a millisecond for these scenes.

In Figure 5.9 we show the performance of our approach with respect to the maximum spherical harmonic order, n, for the various scenes. Depending on the type of scene, the maximum spherical harmonic order can have a significantly effect on the time it takes to compute the spatial audio filter. For scenes where the majority of sources are spherical (Island, Waterfall), the effect is higher as the computational time of analytical projection is directly proportional to square of spherical harmonic order. On the other hand, for scenes which are dominated by box or mesh sources, the effect is significantly smaller as the computational time is dominated by the ray-tracing.

Complexity: Here we derive the computational and memory complexity of our approach with respect to the spherical harmonic order n. The analytical projection for spheres requires evaluating the $n+1$ zonal harmonic coefficients z_l via equation 5.20 where $l \in 0 \ldots n$. Each coefficient involves a sum from $0 \ldots l$, and so the complexity for a single coefficient is $O(l)$. The cost for the projection is then proportional to the sum $\sum_{l=1}^{n+1} l = \frac{(n+1)(n+2)}{2}$. Therefore, the overall computational complexity of the analytical projection is $O(n^2)$. The Monte Carlo approach must evaluate the spherical harmonic basis functions $Y_{lm}(\theta, \phi)$ for each of N rays. For order n, $(n+1)^2$ spherical harmonic coefficients are computed. As a result, the Monte Carlo approach's overall computational complexity is $O(Nn^2)$. The HRTF storage requires $(n+1)^2$ impulse responses of length L. The memory complexity of our method is therefore $O(Ln^2)$.

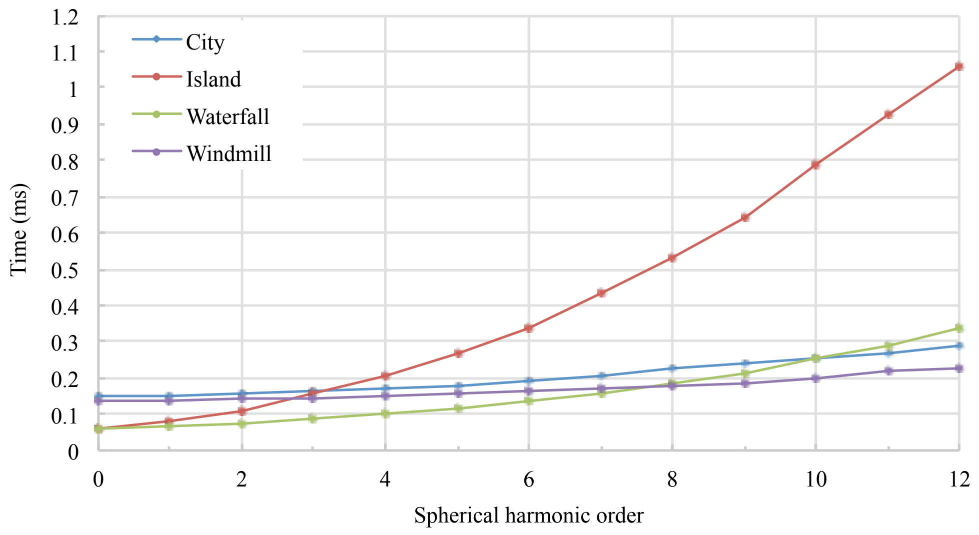


Figure 5.9: The performance of our approach varies with respect to the spherical harmonic order for the four scenes.

5.6.2 Spatial Impulse Response Construction

The capabilities of our spatial impulse response construction algorithm were evaluated on 6 different scenes within the Unity™game engine. The scenes, shown in Figure 5.10, each have 6 to 9 sound sources, some of which are dynamic, and have geometry complexity typical of virtual reality and game environments. The scenes also contain interactive elements like moving doors that impact the resulting auralization.

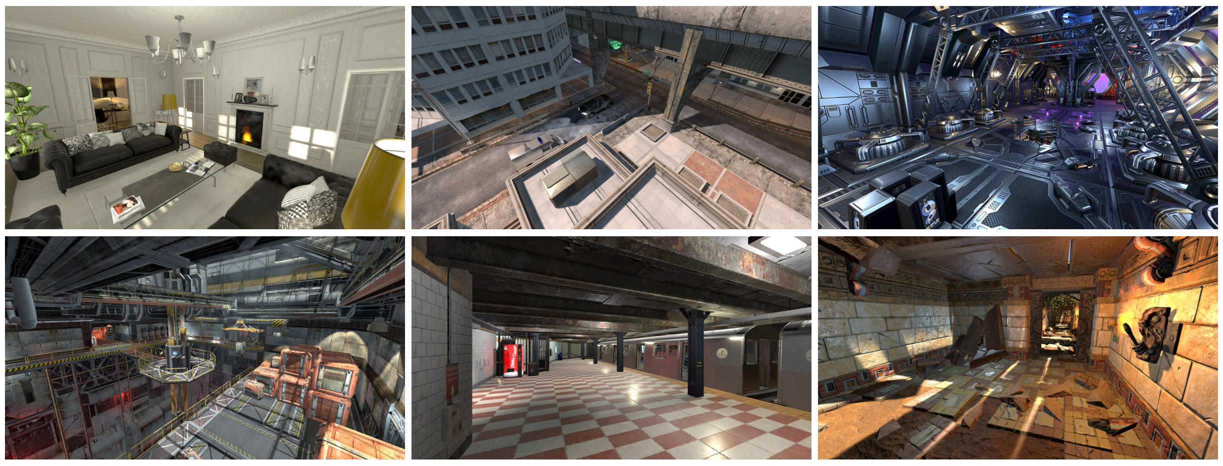


Figure 5.10: The six benchmark scenes used to evaluate our spatial impulse response construction technique. From left to right and top to bottom: apartment, city, hangar, industrial, subway, temple.

The main results of our system on these scenes are summarized in Table 5.3. The times were measured on a 6-core 3.50GHz Intel i7-5930K machine by measuring the average time over the demo sequences in the

	Scene Complexity		**Propagation**	**Rendering**	**Per-path**	**Our technique**		
Scene	# Tri.	# Sources	Time (ms)	(% Real time)	SRIR (ms)	SRIR (ms)	Total (ms)	Speedup
Apartment	491,683	6	30.5	5.3	242.6	28.4	58.9	**8.6**
City	113,388	6	30.3	6.7	488.0	53.6	83.9	**9.1**
Hangar	473,328	7	31.4	7.1	449.9	53.8	85.2	**8.4**
Industrial	202,642	7	29.9	8.1	466.2	68.8	98.7	**6.8**
Subway	125,449	9	30.3	5.5	296.9	44.4	74.7	**6.7**
Temple	48,700	8	30.4	8.3	368.5	51.5	81.9	**7.1**

Table 5.3: The main results of our sound propagation and rendering system for the six benchmark scenes. We report the time taken for sound propagation separately from the SRIR construction time, and we compare the performance of our method to the performance of SRIR construction using per-path HRTF interpolation. Our method provides a speedup of $6.7 - 9.1$ over the previous approach.

supplementary video.

Performance: The overall performance of our algorithm is reported for each scene in Table 5.3. By design, the time taken for sound propagation is about the same for all scenes, roughly 30ms. For the SRIR construction, our approach takes anywhere from 28.4ms to 68.8ms, whereas the previous approach takes 242.6ms to 488.0ms. This significant variation is mostly due to differences in the impulse response length and the number of sources in each scene. Differences in the directivity present in the scenes may also account for some of the variation. When the performance of the per-path approach is compared to ours, we observe a significant $6.7 - 9.1$ times speedup. This speedup enables our approach to update the SRIR fast enough for interactive applications. Our technique is able to satisfy perceptual latency thresholds of around 100ms, whereas the per-path approach is so slow that it introduces noticeable delay.

A significant parameter in the execution time of our algorithm is the spherical harmonic (SH) order at which the directivity is evaluated for each partition, n_{max}. In Figure 5.11 we show how the performance scales with respect to n_{max} for the six scenes. All scenes show a quadratic increase in execution time with respect to the SH order. This characteristic is mostly due to the evaluation of our directivity metric for every partition, not the cost of convolution with the HRTF. This is because our method tends to only use $\tilde{n}$ close to n_{max} for the first several partitions (see Figure 5.12). As a result, the increase in computational cost for the convolution of the RIR with the HRTF is smaller than the increase due to evaluation of the metric.

The primary benefit of our approach is that it enables the spherical harmonic order of the spatial sound to vary according to the directivity present in the room impulse response. This is illustrated in Figure 5.12. Toward the beginning of the impulse response where the directivity is stronger due to the presence of direct

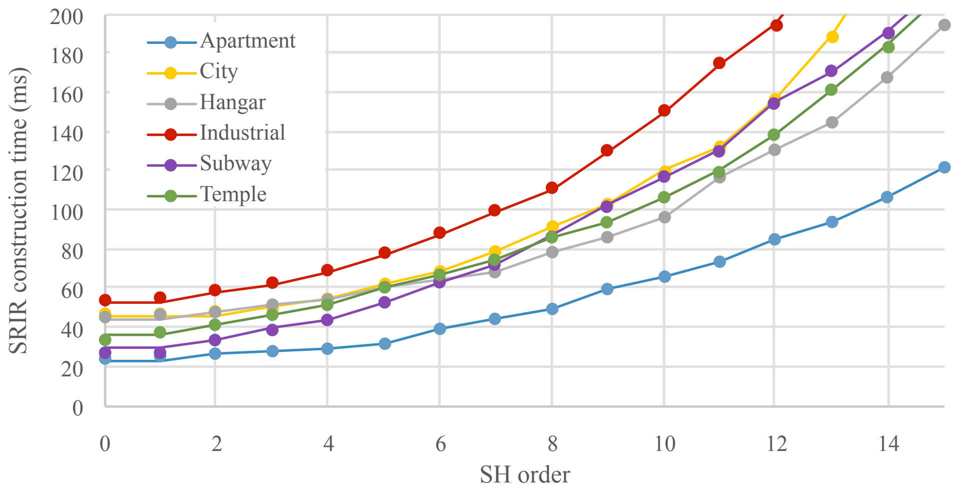


Figure 5.11: The performance of our spatial room impulse response construction algorithm scales quadratically with respect to the maximum spherical harmonic order n_{max}. We used $n_{max} = 4$ to generate the main results of our approach. The differences between the scenes are due to variation in the lengths of the impulse responses and the number of sources.

sound and early reflections, our approach tends to use a higher SH order. The SH order that is required decreases quickly thereafter due to the increasingly diffuse sound field. For the last half of the IR, 1st order is all that is needed to satisfy our directivity metric. This results in a large overall savings in computation versus using a fixed order for the whole impulse response. Our perceptually-based metric keeps the sound quality about the same as doing per-path HRTF interpolation, but has a much better overall performance.

Latency: There are many sources of latency in our system. We enumerate these in Table 5.4 and report an estimate of the total end-to-end latency of the audio pipeline based on the performance on the benchmark scenes. Sound propagation is responsible for roughly 30ms of latency, while SRIR construction can take $30 - 70$ms. There is an additional 10.7ms of latency for updating the convolution system with the new impulse response, while the convolution itself only adds 2.7ms of delay. Finally, a significant amount of latency (21.3ms) is incurred by the lengthy audio device output buffer used by Unity™. The overall latency can range from 82.3ms to 134.9ms and the variation is strongly dependent on the scene. This is around the desired 100ms latency target for interactive audio. On the other hand, the per-path SRIR construction approach has a total latency of around 500ms for most of the scenes. This amount of latency is unacceptable for interactive applications and leads to noticeable delay and artifacts in the audio rendering with dynamic scenes.

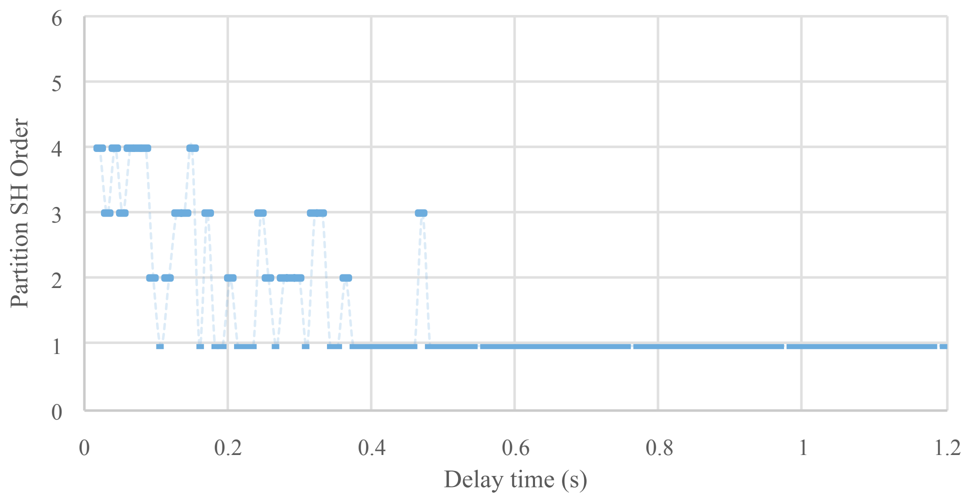


Figure 5.12: The spherical harmonic order $\tilde{n}$ determined by our directivity metric for each partition varies across the impulse response length. We observe a tendency for higher SH order toward the beginning of the IR where the pressure is greater and where there are more distinct paths with strong directivity (e.g. direct sound and early reflections).

Latency Source	Latency (ms)
Sound propagation	29.9 - 31.4
SRIR construction	28.4 - 68.8
Convolution IR update	0 - 10.7
Convolution	2.7
Audio output buffer	21.3
Total	82.3 - 134.9

Table 5.4: The sources of latency in our sound propagation and rendering system. The total latency for our system is around the 100ms latency target needed for interactive virtual reality.

It is important to note that the end-to-end latency of our system could be reduced further by using a shorter audio output buffer of just a few milliseconds (e.g. 64 samples, 1.3ms). The convolution IR update latency could also be reduced to 64 samples by using shorter FFT blocks for convolution with the source's audio, though this would decrease the performance of the convolution for long IRs.

5.7 User Evaluation

In this section we present two user studies that were conducted to evaluate the impact of our spatial sound techniques and to perform comparisons to previous techniques. In the first study, we investigate the impact of using our analytical-monte carlo technique for area and volume sound sources and perform a comparison to a naïve point-sampling approach. In the second study, we evaluate our technique for spatial impulse response construction and perform a comparison with a previous technique based on per-path HRTF interpolation.

5.7.1 Evaluation of Area and Volume Sources

We have conducted a user evaluation to study the effect of area-volumetric sound sources on subjective preference of users in virtual environments. We compare the sound generated by a point-sampling technique, called the *base* method, with the sound generated by the Analytical-Monte Carlo technique (Section 5.3), called *our* method. In the point-sampling approach, we represent an area-volumetric source with a collection of discrete point sources. The number of point sources used to represent the area-volumetric source was chosen to ensure that the runtime computational requirements of the point-sampling technique matched our Analytical-Monte Carlo technique.

Study Design: The study uses a within-subject experiment design with an A-B session comparison protocol. The study has two comparison conditions: *base* vs. *our*, and *our* vs. *base*. Corresponding to each condition, a pair of VR sessions was generated with identical visual rendering techniques but with different spatial audio techniques. These two comparisons conditions were produced for each of 3 scenes (Island, Waterfall, Windmill), for a total of 6 scenarios. These 6 scenarios were presented to the participant in a random order. The participant was unaware as to which VR session (A and B) corresponded to which technique (*base* and *our* method).

The virtual avatar for the participant was spawned at a position in each scenario and the participant was free to move and rotate their head. The position and orientation of the participant's head were tracked by the head-mounted display and the audio and visuals were updated correspondingly. Each scenario would last for one minute and the participant had the ability to toggle between the two sessions as many times as he/she wants. After completion of each scenario, the participant answered the following subjective questionnaire:

1. In which session did the spatial extent of the sound better match the visuals?
2. In which session did you feel most enveloped by the soundscape?
3. Which session did you prefer?

The responses were recorded on a 5-point Likert scale: 5 meant strong preference for Session A, 3 meant no preference and 1 meant strong preference for session B.

System Details: Visual information was presented to the participants via the Oculus Rift DK2™head-mounted display (HMD). Sound was produced through the Sennheiser HD 700 open-ear headphones. We used the standard KEMAR HRTF dataset for auralization. The head position and orientation given by the HMD was used to update both the visual rendering and the spatial audio rendering.

Procedure: The study was conducted on a total of 30 participants, all between the age of 23 and 49. There were 23 males and 7 females, and the mean age of the group is 33 years. All participants had normal or corrected-to-normal vision. Participants were given a basic hearing test with conventional audiometry equipment (e.g. audiometer) to detect their hearing thresholds. Participants with a hearing threshold within 20 dB of audiometric zero were included in the study. All participants passed the hearing test. The study was conducted in-person in a single session lasting 30-45 minutes. Before the experiment, participants filled a background questionnaire and were given detailed instructions. The participants were also trained on how to wear and use the equipment and were given one trial to get acquainted with the system. Then, participants were presented the 6 scenarios in a random order and asked to rate their preference after each scenario. Participants were allowed to take a break at any time if desired. After all the 6 scenarios, the experiment was completed. All the subjects completed the study.

Research Hypothesis: The research hypotheses of this study were: 1) The proposed technique improves audio-visual spatial extent match, sense of envelopment by soundscape, and general preference, in VR environments compared to the point-sampling technique. 2) The amount of improvement depends on the type of scene and the type of area-volumetric sound sources.

Results: Figure 5.13 shows the results of our user study. The scores of the *base vs. our* condition were reversed and combined with the *our vs. base* condition. The comparison score is averaged over all the participants for each of the three questions and three scenes. A score less than 3 indicates a preference for point-sampling technique whereas a score greater than 3 indicates a preference for our Analytical-Monte Carlo technique. A score of 3 indicates no preference. Our Analytical-Monte Carlo technique performed

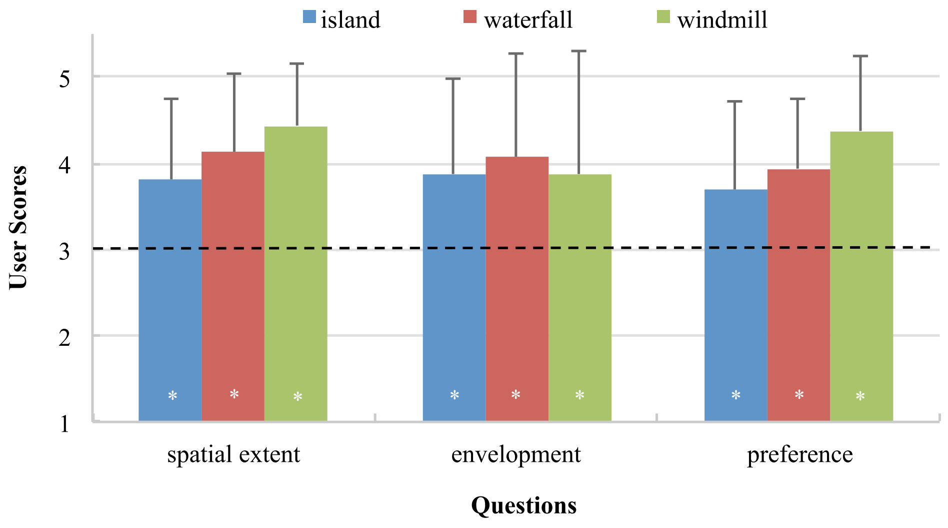


Figure 5.13: User evaluation results for the subjective questionnaire. The comparison score is averaged over all the subjects and is plotted for each question and scene. A score of 1 represents a strong preference for the point-sampling technique and A score of 5 represents strong preference for our analytical-monte carlo technique. The horizontal dashed line presents a score of 3 indicating no preference between the two techniques. Standard deviation is represented by the error bars. The symbol * denotes significance levels of $p < 0.001$.

better than the point-sampling technique for all the questions and all the scenes. For the spatial extent question, the mean scores were 3.5, 3.9 and 4.3 for the island, waterfall, and windmill scene, respectively. With regards to sense of envelopment, the mean scores were 3.6 (island), 4.0 (waterfall), and 4.4 (windmill). As for subjective preference, the mean scores were 3.6, 4.0, and 4.5 for the island, waterfall, and windmill scenes, respectively. These results are statistically significant for all the questions and all the scenes (one-sampled Wilcoxon signed-rank test, one-tailed, $p < 0.001$). Along with independent Wilcoxon tests, one-way Kruskal-Wallis test was also performed for each question across the different scenes: spatial extent (H(2)=10.43806, p=0.00541), envelopment (H(2)=8.17093, p=0.01682), and preference (H(2)=11.66496, p=0.00293). This shows statistical significance of the results for all the three questions across the different types of scenes. These results demonstrate that participants perceive better match in the audio-visual spatial extent (of the area-volumetric source), increased sense of envelopment by the soundscape and higher preference with our technique compared to the point-sampling technique. Additionally, the amount of improvement depends on the scene type and the type of area-volumetric sound sources present in the scene.

5.7.2 Evalution of Spatial Impulse Response Construction

In order to evaluate the subjective impact of our spatial sound approach, we carried out a user evaluation in interactive virtual reality environments. The study compared the sound generated by our perceptually-based SRIR construction technique (Section 5.4), called *our* method, to the per-path HRTF interpolation approach, called the *base* method. For the *base* method, we tested two configurations: $base_s$, a single-threaded implementation (update time, $250 - 500$ms), and $base_p$, a parallel implementation where 10x as many threads are used to compute the SRIR (update time, $25 - 50$ms). Therefore, $base_p$ has about the same total latency as *our* approach, but uses 10x as much compute power, while $base_s$ has a latency about 10x that of both other methods.

The hypotheses of this study were: 1) $base_s$ has a much higher latency than *our* method and so there will be a preference for *our method*. 2) There will be no preference between $base_p$ and *our* method because the latency is similar and the sound is perceptually indistinguishable. 3) The strength of preference for either method will be dependent on the type of scene.

Study Design: The study was implemented using a within-subject experiment design and an A-B comparison protocol. Four different comparison conditions were evaluated: $base_s$ vs. *our*, *our* vs. $base_s$, $base_p$ vs. *our*, and *our* vs. $base_p$. These conditions were tested for 3 different scenes (City, Industrial, Temple), resulting in 12 different scenarios. Each scenario was repeated twice during the experiment, so each participant experienced a total of 24 trials. The trials were presented in a random order in two sets of 12 with a short break in-between. In each of these trials, the participant was presented an interactive audio-visual virtual reality experience where the sound was generated using either method A or method B according to the current condition under evaluation. During a trial, which lasted one minute, the participant was free to toggle between method A and B as many times as they wanted. The participant was spawned in the scene at a static position where they were able to freely move their head. The head movement was tracked using the head-mounted display and used to update the orientation of the listener in the virtual environment.

After each trial was completed, the participant answered a short subjective questionnaire to indicate their preferences on a 5-point Likert scale with respect to the following questions:

1. In which mode did the audio better correspond to the visuals?

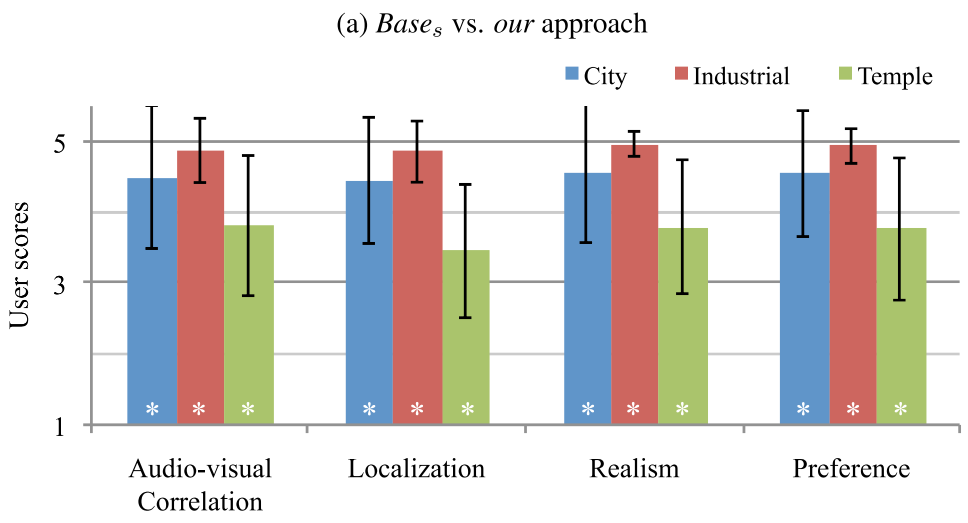
2. In which mode could you better localize the sound?
3. Which mode was more realistic?
4. Which mode did you prefer?

A response of 1 indicated a strong preference for method A, while a response of 5 indicated a strong preference for method B. A response of 3 means that the subject had no preferences.

System Details: The visual display of the virtual reality environment was presented using an Oculus Rift CV1™head-mounted display. The audio was delivered through the headphones that are integrated into the display. The study used a 14-core 2.60GHz Intel Xeon E5-2697v3 machine in order to render the audio for the $base_p$ case without glitches. The scenes were also simplified to contain just 1 or 2 sound sources so that the $base_p$ case would run in real time. A diffuse-field equalized version of the HRTF of subject 36 from the ARI HRTF database was used for all subjects (Acoustics Research Institute of the Austrian Academy of Sciences, 2016).

Results: The questionnaire responses of the user evaluation are summarized in Figure 5.14. There was a total of 16 subjects who completed the study. The scores for the *our* vs. $base_s$ and *our* vs. $base_p$ conditions were reversed and combined with the scores for $base_s$ vs. *our* and $base_p$ vs. *our*, respectively. Subjects tended to answer all four questions with the same answer, so there is not much variation in responses among the questions.

For the comparison between $base_s$ and *our* method, the mean scores for all questions are between 4.45 and 4.97 for the City and Industrial scenes. These scenes contain fast-moving sound sources and so produce very noticeable delay or jumpiness when the sound is generated using the slow $base_s$ method. For the Temple scene, the preference for our method is slightly less, with scores on all questions ranging from 3.45 to 3.80. This difference could be because the dynamic element in the Temple scene, an opening/closing door, moves slower than the sound sources in the other scenes. As a result, the latency differences between the methods are less noticeable in that scene. When analyzed with a one-sampled one-tailed Wilcoxon signed-rank test ($p < 0.001$) these results show a statistically significant preference for *our* method over $base_s$ for all scenes and questions. This confirms our first study hypothesis that *our* method will be preferred over the



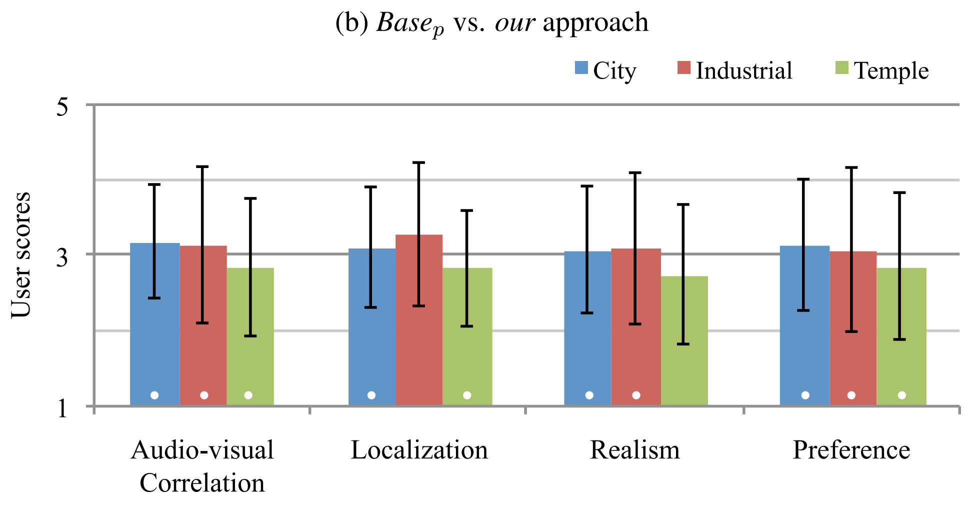


Figure 5.14: The results of the user evaluation of our technique. We report the average response for each question and scene, where a value of 1 indicates a strong preference for the first method (*Base$_s$* or *Base$_p$*), a value of 5 indicates a strong preference for *our* method, and a value of 3 indicates no preference. The error bars correspond to the standard deviation. The * symbol indicates a significance with $p < 0.001$, while the • symbol indicates a lack of statistical significance ($p \geq 0.05$).

base$_s$ method. The differences between the Temple and Industrial/City scenes also supports our third study hypothesis that the preference will be dependent on the type of scene.

The other study comparison was between *our* method and *base$_p$*, the parallel version of the per-path SRIR construction technique. For this case, the mean scores for all scenes and questions are clustered between 2.75 and 3.28. A one-sampled two-tailed Wilcoxon signed-rank test ($p < 0.05$) was used to determine if there was any significant preference for either method. For all but two cases there is no preference between the methods ($p \geq 0.05$). For the Industrial scene, there is a small preference for *our* method on the localization question

($p = 0.026$), while for the Temple scene there is a small preference for the *base*$_p$ method ($p = 0.035$) on the realism question. Overall, this confirms our second hypothesis that the differences between *base*$_p$ and *our* method are not noticeable when the latency of the per-path SRIR construction is reduced by using 10x as many CPU threads.

5.8 Conclusion, Limitations, and Future Work

In this chapter we have presented two techniques for efficient low-latency computation of spatial sound for sound propagation. The first contribution is an approach that computes the spatial sound filter for area and volume sources in the spherical harmonic domain using either an analytical or Monte Carlo projection integral. This gives a speedup of $2 - 3$ orders of magnitude over a naïve point-sampling approach. The second contribution is a technique for efficiently computing spatial sound for sound propagation using a perceptual directivity metric that is applied to impulse response partitions in order to save computation. This method gives a speedup of $6.7 - 9.1$ over the previous algorithm for spatial impulse response construction. For the first time, these improvements enable spatial sound to be computed with low-latency for area sources and interactive sound propagation.

Limitations and Future Work: Our approach has some limitations. Since we perform a single projection into the spherical harmonic domain for each area source shape, we assume the sound of each shape is delayed equally, rather than in a directionally-dependent manner across the shape. This results in loss of phase information due to different propagation path lengths. Our technique is not very efficient for sharp directivities due to the choice of spherical harmonics as the basis functions. These basis functions are not ideal for sharp projections and can result in high-order expansions with increased computational cost. Since our technique is valid for any orthonormal basis functions of the spherical domain, other more appropriate basis functions like the spherical wavelets can be used for sharp projection functions. However, recent work has suggested that a 4th-order spherical harmonic HRTF representation may be sufficient for accurate localization, and that most HRTF features are accurately captured at order 14 or less (Romigh et al., 2015). As a result, it may not be perceptually necessary to compute sharp directivities. Another limitation is aliasing of thin sources. In case of the Monte Carlo area source projection, thin sources (e.g. a line) can lead to aliasing if none of the random rays intersect the source. This problems can be ameliorated, though not eliminated, by increasing the number of rays traced. We would like to explore more efficient formulations for such types of sources. In addition, our current analytical approach cannot handle occlusion effects from obstacles in the scene, since it assumes the entire sphere is visible. We would like to add an occlusion factor into our projection function to incorporate this. Since our IR directivity metric is applied to partitions, it is possible that the partitions may not be of sufficient resolution to capture variation in sound directivity at time scales less than one partition. This can be ameliorated by reducing the partition size, though this will result in a greater expense during convolution with the HRTF (Section 5.4.2) because more smaller FFTs must be evaluated. Using a smaller partition size also has the drawback that more propagation paths are required to reduced the noise in the Monte Carlo directivity estimation (5.27).

With regards to future work, the first improvement we would like to make is to integrate source directivity in our area source formulation. The Monte-Carlo projection method can easily handle frequency-independent source directivity by applying an additional gain coefficient to equations (5.24) and (5.25). Each vertex in the source mesh can be given a radiation strength that is used to generate simple source directivity. More complex frequency-dependent radiation patterns can be efficiently achieved by performing the projection independently for multiple frequency bands. In a preprocessing step, the HRTF would be filtered into the same frequency bands. Then, the filters for all bands are constructed independently and summed to generate the final HRTF filter that incorporates frequency-dependent source directivity. Another avenue of future work is to perform a detailed evaluation to quantify the effect that sound propagation (as opposed to only direct sound) has on latency detection thresholds for interactive spatial sound.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

In this dissertation, we have presented techniques that enable interactive simulation of geometric sound propagation in dynamic environments. These techniques solve various challenges faced by previous interactive, geometric sound propagation approaches. These include reducing the number of rays required, automatically determining acoustic materials, and efficiently computing spatial sound for sound propagation. We have evaluated the performance on many varied environments and observe significant improvements in sound propagation and spatial sound over previous methods. The accuracy of the proposed techniques has been validated by comparison to real-world measurements, and also through the subjective evaluation of the results by user studies. The results of the proposed techniques demonstrate significant improvement over the previous state of the art, thereby enabling the interactive simulation of complex multi-source dynamic scenes that was not possible using previous techniques.

6.1 Summary of Results

We have proposed several novel algorithms that use temporal coherence to improve the performance of sound propagation. These include the *specular path cache* for specular reflections, the *diffuse path cache* for diffuse reflections, the *impulse response cache* for impulse responses, and an automatic technique for adaptively determine the impulse response length. These algorithms use sound propagation information from previous frames to reduce the computation needed on future frames. Through the use of these techniques, the number of rays traced during sound propagation can be greatly reduced while maintaining a similar level of sound quality. This provides over an order of magnitude speedup when applied to the ray-based image source method or Monte Carlo path tracing. As a result of these temporal coherence techniques, sound propagation can be computed for large, complex, dynamic scenes with many sources at interactive rates.

Secondly, we presented an approach for automatically determining the acoustic material properties of a real environment. Our technique uses a two-step approach. In the first step, we use computer vision algorithms and deep learning to assign material categories to the reconstructed 3D surfaces of the environment.

In the second step, we improve the accuracy of the materials with a novel iterative optimization algorithm based on a least squares solver. The optimization approach uses measured impulse responses from the real environment as the optimization target. On each iteration, the algorithm solves for the materials that produce simulated impulse responses that closely match the measured data. With this material classification and optimization approach, acoustic materials can be estimated automatically without any time-consuming manual intervention. We evaluated the subjective differences between the measured IRs and the optimized simulated IRs on several rooms and observed no significant differences. As a result, new applications of interactive sound propagation are possible, such as for augmented reality or teleconferencing where it would otherwise be too difficult or slow to acquire the material properties.

Finally, we proposed techniques for the efficient computation of low-latency spatial sound for interactive sound propagation. Using a formulation based on projection into the spherical harmonic domain, we showed how spatial sound filters can be computed for area and volume sound sources. The complexity of our approach scales with the projected area, not the actual size of the sound sources. Therefore, it can handle very large sound sources such as an ocean, lake, or river in under a millisecond. We also presented a novel perceptually-based algorithm for efficient construction of the spatial impulse response. This approach performs convolution of the HRTF with the impulse response in the spherical harmonic domain, and adaptively determines the SH order for each impulse response partition based on a perceptual error threshold. As a result, our technique can compute a spatial impulse response almost an order of magnitude faster than previous approaches.

6.2 Limitations

In this section, we discuss the limitations of the proposed techniques. Since we use geometric algorithms to compute sound propagation, the standard limitations of those approaches apply. Namely, our techniques are less accurate for low frequencies or when the size of surface primitives is very small.

The temporal coherence techniques are based on using cached information from the past to improve the sound propagation quality. If there is a large sudden change in a scene, such as a door slamming shut, the cached data may be out of date and invalid. It can take a short time for the temporal coherence techniques to adapt to this sort of change, causing minor audio artifacts such as where a sound source is briefly audible that should not be. This lag is controllable via the parameter τ that trades interactivity for sound quality. It is also possible to reset the cache if there is a sudden change in the scene, thereby entirely avoiding artifacts.

The proposed *diffuse path cache* approach also uses a subdivision of surfaces to group ray paths together. If the subdivision is too coarse, it can introduce small errors in the sound propagation paths, particularly in the delay time. Conversely, if the subdivision is small, the improvement in quality due to the cache will be less. Our adaptive IR length algorithm is based on an analytical model of the human threshold of hearing that may not work for all listeners. Individualized measurements can be used to overcome this limitation.

The automatic acoustic material classification uses computer vision algorithms to acquire visual data and to classify the material categories that are present in the scene. In many cases, the material categories that are visually salient are different than the ones that are important for acoustic simulation. The visual appearance of a material is sometimes independent of the acoustic attributes, and this can lead to incorrect material assignments. Therefore, the accuracy of the classification approach is strongly dependent on the scene, the materials present, the categories that were trained for the deep learning models, and the quality and resolution of the capture system. The optimization approach attempts to overcome some of these shortcomings by modifying the initial material properties to better match measurements. However, the optimization has many degrees of freedom and is generally under constrained. Therefore, it is possible that the optimization may not converge to a global minimum, and the optimized materials may not exactly match the actual material properties.

The techniques for low-latency spatial sound are based on representing the HRTF using the spherical harmonic basis functions. Therefore, their accuracy depends on the spherical harmonic order that is used. If too low an order is used, strongly directional sounds are not represented accurately. This can negatively impact localization performance in some cases. However, in the case of our spatial impulse response construction algorithm, we use a perceptual threshold to determine the SH order. Therefore, any error introduced by using low spherical harmonics should be imperceptible. During spatial IR construction, we evaluate the directivity and SH order for each IR partition. This can introduce error if the directivity varies strongly within a partition. For sound sources that have a small projected area, our approach for area/volume sources may perform poorly since rays are unlikely to hit the source. Our formulations do not consider the distance-dependence of the HRTF. As a result, they do not reproduce any near-field HRTF effects.

6.3 Future Work

There are many ways that the proposed techniques could be improved. In the context of temporal coherence, we would like to investigate the use of feedback from the sound propagation simulation to automatically detect when the sound field has significantly changed. If there was a large change, the smoothing time parameter τ could be reduced automatically to avoid the smoothing artifacts inherent in our *diffuse path cache* and *impulse response cache* approaches. Another possible avenue for future work is to examine how other psychoacoustic phenomena such as auditory masking could be used to improve our adaptive impulse response length approach. Since a loud sound source might mask another quieter source, we could use masking to detect perceptually inaudible sources that would be considered audible using the current approach. This can further reduce the computation needed in scenes with many sources. We would also like to investigate how temporal coherence could be applied to the computation of diffraction, a significant bottleneck. If diffraction could be evaluated probabilistically, e.g. in a Monte Carlo path tracing framework, a significant performance boost could be achieved in combination with the proposed temporal coherence techniques.

For our material classification and optimization approach, we would like to increase the number of material categories that are considered during the visual recognition phase. This would involve training new convolutional neural network models in order to identify more categories that are relevant to acoustic simulation in real environments. We would also like to investigate the use of different audio/visual sensors and to evaluate the impact on the quality of the classification results. It is currently time consuming to measure impulse responses in real-world scenes and it requires specialized equipment. We would like to develop new audio capture techniques that could acquire impulse responses in a user-friendly manner. This would be important for application of our optimization technique to augmented reality applications in the wild. Another avenue for future work would be to investigate optimization approaches other than least squares that would be more likely to converge to a globally optimal solution. We would also like to try integrating our approach into a mobile augmented reality device with integrated sensing (e.g. the Microsoft Hololens™) in order to evaluate the quality of the results in an augmented environment.

While the spherical harmonic basis functions work well as a domain for spatial sound computation, we would like to investigate other orthogonal basis functions such as the spherical wavelets that may perform better with sharp directivities. We would like to investigate different sampling techniques for the area/volume source method that can handle thin sources efficiently. We also want to try sound synthesis techniques to generate different sound from different parts of a large area source. This could reduce unnatural comb filtering that occurs if the same audio is used for the entire area source. For the spatial impulse response construction, we would like to try different partitioning schemes and explore if the quality or performance could be improved. We would also like to investigate if auditory masking may be able to reduce the SH order used for IR construction. Finally, we want to modify our formulations so that they handle near-field HRTF effects.

BIBLIOGRAPHY

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

Acoustics Research Institute of the Austrian Academy of Sciences (2016). The ARI HRTF database. *http://www.kfs.oeaw.ac.at/hrtf, accessed 2016-09-15*.

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

Antani, L. and Manocha, D. (2013). Aural proxies and directionally-varying reverberation for interactive sound propagation in virtual environments. *IEEE transactions on visualization and computer graphics*, 19(4):567–575.

Barron, M. F. (1974). *The effect of early reflections on subjective acoustical quality in concert halls*. PhD thesis, University of Southampton.

Battenberg, E. and Avizienis, R. (2011). Implementing real-time partitioned convolution algorithms on conventional operating systems. In *Proceedings of the 14th International Conference on Digital Audio Effects. Paris, France*.

Begault, D. R. (1996). Audible and inaudible early reflections: thresholds for auralization system design. In *Audio Engineering Society Convention 100*. Audio Engineering Society.

Begault, D. R., Wenzel, E. M., and Anderson, M. R. (2001). Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. *Journal of the Audio Engineering Society*, 49(10):904–916.

Bell, S., Upchurch, P., Snavely, N., and Bala, K. (2015). Material recognition in the wild with the materials in context database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 00:3479–3487.

Bendat, J. S. and Piersol, A. G. (1986). The hilbert transform. *Random Data: Analysis and Measurement Procedures, Fourth Edition*, pages 473–503.

Blauert, J. (1997). Spatial hearing: the psychoacoustics of human sound localization. *MIT Press*, 5:210–212.

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

Branch, J., Prieto, F., and Boulanger, P. (2006). Automatic hole-filling of triangular meshes using local radial basis function. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 727–734. IEEE.

Brown, R. G. (1956). Exponential smoothing for predicting demand. In *Proceedings of the 10th national meeting of the Operations Research Society of America, San Francisco*.

Brungart, D. S., Simpson, B. D., and Kordik, A. J. (2005). The detectability of headtracker latency in virtual audio displays. In *Int. Conf. on Auditory Display*. Georgia Institute of Technology.

Brungart, D. S., Simpson, B. D., Mckinley, R. L., Kordik, A. J., Dallman, R. C., and Ovenshire, D. A. (2004). The Interaction Between Head-Tracker Latency, Source Duration, and Response Time in the Localization of Virtual Sound Sources. In *Int. Conf. on Auditory Display*. Georgia Institute of Technology.

Carvalho, A. P. O. (1995). The use of the sabine and eyring reverberation time equations to churches. *The Journal of the Acoustical Society of America*, 97(5):3319–3319.

Chandak, A., Lauterbach, C., Taylor, M., Ren, Z., and Manocha, D. (2008). Ad-frustum: Adaptive frustum tracing for interactive sound propagation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1707–1722.

Christensen, C. L., Koutsouris, G., and Rindel, J. H. (2014). Estimating absorption of materials to match room model against existing room using a genetic algorithm. In *Forum Acusticum 2014, At Krakow, Poland*.

Christensen, C. L. and Rindel, J. H. (2005). A new scattering method that combines roughness and diffraction effects. In *Forum Acousticum, Budapest, Hungary*.

Ciskowski, R. D. and Brebbia, C. A. (1991). *Boundary element methods in acoustics*. Springer.

Dou, M., Guan, L., Frahm, J.-M., and Fuchs, H. (2013). Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera. In *Computer Vision-ACCV 2012 Workshops*, pages 94–108. Springer.

Egan, M. D. (1988). *Architectural Acoustics*. McGraw-Hill Custom Publishing.

Embrechts, J. J. (2000). Broad spectrum diffusion model for room acoustics ray-tracing algorithms. *The Journal of the Acoustical Society of America*, 107(4):2068–2081.

Evans, M. J., Angus, J. A., and Tew, A. I. (1998). Analyzing head-related transfer function measurements using surface spherical harmonics. *The Journal of the Acoustical Society of America*, 104(4):2400–2411.

Fletcher, H. (1940). Auditory patterns. *Reviews of modern physics*, 12(1):47.

Foster, S. (1986). Impulse response measurement using golay codes. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 929–932. IEEE.

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

Gardner, W. G. (1994). Efficient convolution without input/output delay. In *Audio Engineering Society Convention 97*. Audio Engineering Society.

Gardner, W. G. (2002). Reverberation algorithms. In *Applications of digital signal processing to audio and acoustics*, pages 85–131. Springer.

Green, R. (2003). Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference*, volume 56.

Härmä, A., Jakka, J., Tikander, M., Karjalainen, M., Lokki, T., Hiipakka, J., and Lorho, G. (2004). Augmented reality audio for mobile and wearable appliances. *Journal of the Audio Engineering Society*, 52(6):618–639.

Hendrix, C. and Barfield, W. (1996). The sense of presence within auditory virtual environments. *Presence: Teleoperators & Virtual Environments*, 5(3):290–301.

ISO (2003). ISO 354, Acoustics—Measurement of sound absorption in a reverberation room. *International Standards Organisation*, (354).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.

Keller, J. B. (1962). Geometrical theory of diffraction. *JOSA*, 52(2):116–130.

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

Krokstad, A., Strom, S., and Sørsdal, S. (1968). Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibration*, 8(1):118–125.

Kulkarni, A., Isabelle, S., and Colburn, H. (1995). On the minimum-phase approximation of head-related transfer functions. In *Applications of Signal Processing to Audio and Acoustics, 1995., IEEE ASSP Workshop on*, pages 84–87. IEEE.

Kuttruff, H. (2007). *Acoustics: An Introduction*. CRC Press.

Kuttruff, K. H. (1993). Auralization of impulse responses modeled on the basis of ray-tracing results. *Journal of the Audio Engineering Society*, 41(11):876–880.

Larsson, P., Väljamäe, A., Västfjäll, D., Tajadura-Jiménez, A., and Kleiner, M. (2010). Auditory-induced presence in mixed reality environments and related technology. In *The Engineering of Mixed Reality Systems*, pages 143–163. Springer.

Lentz, T., Schröder, D., Vorländer, M., and Assenmacher, I. (2007). Virtual reality system with integrated sound field simulation and reproduction. *EURASIP journal on applied signal processing*, 2007(1):187–187.

Lindau, A. (2009). The perception of system latency in dynamic binaural synthesis. *Proc. of 35th DAGA*, pages 1063–1066.

Mehra, R., Antani, L., Kim, S., and Manocha, D. (2014). Source and listener directivity for interactive wave-based sound propagation. *Visualization and Computer Graphics, IEEE Transactions on*, 20(4):495–503.

Møller, H. (1992). Fundamentals of binaural technology. *Applied acoustics*, 36(3):171–218.

Monks, M., Oh, B. M., and Dorsey, J. (2000). Audioptimization: goal-based acoustic design. *Computer Graphics and Applications, IEEE*, 20(3):76–90.

Mückl, G. and Dachsbacher, C. (2014). Precomputing sound scattering for structured surfaces. In *Proceedings of the 14th Eurographics Symposium on Parallel Graphics and Visualization*, pages 73–80.

Müller-Tomfelde, C. (2001). Time varying filter in non-uniform block convolution. In *Proc. of the COST G-6 Conference on Digital Audio Effects*.

Nava, G.-P. (2006). *Inverse sound rendering: In-situ estimation of surface acoustic impedance for acoustic simulation and design of real indoor environments*. PhD thesis, University of Tokyo.

Nehab, D., Sander, P. V., Lawrence, J., Tatarchuk, N., and Isidoro, J. R. (2007). Accelerating real-time shading with reverse reprojection caching. In *Graphics hardware*, volume 41, pages 61–62.

Nosal, E.-M., Hodgson, M., and Ashdown, I. (2004). Improved algorithms and methods for room sound-field prediction by acoustical radiosity in arbitrary polyhedral rooms. *The Journal of the Acoustical Society of America*, 116(2):970–980.

Ondet, A. and Barbry, J. (1989). Modeling of sound propagation in fitted workshops using ray tracing. *The Journal of the Acoustical Society of America*, 85(2):787–796.

Pulkki, V. (1997). Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 45(6):456–466.

Pulkki, V. et al. (2001). *Spatial sound generation and perception by amplitude panning techniques*. Helsinki University of Technology.

Rafaely, B. and Avni, A. (2010). Interaural cross correlation in a sound field represented by spherical harmonics. *The Journal of the Acoustical Society of America*, 127(2):823–828.

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

Raghuvanshi, N. and Snyder, J. (2014). Parametric wave field coding for precomputed sound propagation. *ACM Transactions on Graphics (TOG)*, 33(4):38.

Rindel, J. H. and Christensen, C. L. (2007). Odeon, a design tool for noise control in indoor environments. In *Proceedings of the International Conference Noise at work. Lille*, pages 1–9.

Robinson, D. and Dadson, R. (1957). Threshold of hearing and equal-loudness relations for pure tones, and the loudness function. *The Journal of the Acoustical Society of America*, 29(12):1284–1288.

Romigh, G., Brungart, D., Stern, R., and Simpson, B. (2015). Efficient real spherical harmonic representation of head-related transfer functions. *IEEE Journal of Selected Topics in Signal Processing*, 9(5).

Sabine, W. C. (1922). *Collected papers on acoustics*. Harvard university press.

Saksela, K., Botts, J., and Savioja, L. (2015). Optimization of absorption placement using geometrical acoustic models and least squares. *The Journal of the Acoustical Society of America*, 137(4):EL274–EL280.

Sandvad, J. (1996). Dynamic aspects of auditory virtual environments. In *Audio Engineering Society Convention 100*. Audio Engineering Society.

Savioja, L. (2010). Real-time 3d finite-difference time-domain simulation of low-and mid-frequency room acoustics. In *13th International Conference on Digital Audio Effects (DAFx-10)*, volume 1, pages 77–84.

Scherzer, D., Jeschke, S., and Wimmer, M. (2007). Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 45–50. Eurographics Association.

Scherzer, D., Wimmer, M., Eisemann, E., et al. (2011). A survey on temporal coherence methods in real-time rendering.

Schissler, C., Loftin, C., and Manocha, D. (2017a). Acoustic classification and optimization for multi-modal rendering of real-world scenes. *IEEE Transactions on Visualization and Computer Graphics*.

Schissler, C. and Manocha, D. (2011). Gsound: Interactive sound propagation for games. In *Audio Engineering Society Conference: 41st International Conference: Audio for Games*. Audio Engineering Society.

Schissler, C. and Manocha, D. (2016a). Adaptive impulse response modeling for interactive sound propagation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 71–78. ACM.

Schissler, C. and Manocha, D. (2016b). Interactive sound propagation and rendering for large multi-source scenes. *ACM Transactions on Graphics (TOG)*, 36(1).

Schissler, C., Mehra, R., and Manocha, D. (2014). High-order diffraction and diffuse reflections for interactive sound propagation in large environments. *ACM Transactions on Graphics (SIGGRAPH 2014)*, 33(4):39.

Schissler, C., Nicholls, A., and Mehra, R. (2016). Efficient hrtf-based spatial audio for area and volumetric sources. *IEEE Transactions on Visualization and Computer Graphics*.

Schissler, C., Stirling, P., and Mehra, R. (2017b). Efficient construction of the spatial room impulse response. In *Virtual Reality (VR), 2017 IEEE*, pages 122–130. IEEE.

Schröder, D. (2011). *Physically based real-time auralization of interactive virtual environments*, volume 11. Logos Verlag Berlin GmbH.

Schröder, D., Dross, P., and Vorländer, M. (2007). A fast reverberation estimator for virtual environments. In *Audio Engineering Society Conference: 30th International Conference: Intelligent Audio Environments*. Audio Engineering Society.

Schroeder, M. R. (1961). Natural sounding artificial reverberation. In *Audio Engineering Society Convention 13*. Audio Engineering Society.

Seddeq, H. S. (2009). Factors influencing acoustic performance of sound absorptive materials. *Aust. J. Basic Appl. Sci*, 3(4):4610–7.

Shirley, P. and Wang, C. (1994). Direct lighting calculation by monte carlo integration. In *Photorealistic Rendering in Computer Graphics*, pages 52–59. Springer.

Sloan, P.-P. (2013). Efficient spherical harmonic evaluation. *Journal of Computer Graphics Techniques*, 2(2):84–90.

Stephenson, U. M. (2010). An energetic approach for the simulation of diffraction within ray tracing based on the uncertainty relation. *Acta Acustica united with Acustica*, 96(3):516–535.

Strauss, H. (1998). Implementing doppler shifts for virtual auditory environments. In *Audio Engineering Society Convention 104*. Audio Engineering Society.

Svensson, U. P., Fred, R. I., and Vanderkooy, J. (1999). An analytic secondary source model of edge diffraction impulse responses. *The Journal of the Acoustical Society of America*, 106(5):2331–2344.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tang, Y. (2013). Deep learning using linear support vector machines. In *ICML Workshop on Challenges in Representation Learning*, pages 1–6.

Taylor, M., Chandak, A., Antani, L., and Manocha, D. (2009). Resound: interactive sound rendering for dynamic virtual environments. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 271–280. ACM.

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

Terhardt, E. (1979). Calculating virtual pitch. *Hearing research*, 1(2):155–182.

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

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

Wald, I., Woop, S., Benthin, C., Johnson, G. S., and Ernst, M. (2014). Embree: a kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG)*, 33(4):143.

Wang, J. and Oliveira, M. M. (2003). A hole-filling strategy for reconstruction of smooth surfaces in range images. In *Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on*, pages 11–18. IEEE.

Weinmann, M. and Klein, R. (2015). Advances in geometry and reflectance acquisition. In *SIGGRAPH Asia 2015 Courses*, SA '15, pages 1:1–1:71.

Wenzel, E. M., Miller, J. D., and Abel, J. S. (2000). A software-based system for interactive spatial sound synthesis. In *ICAD, 6th Intl. Conf. on Aud. Disp*, pages 151–156.

Zotkin, D. N., Duraiswami, R., Gumerov, N., et al. (2009). Regularized hrtf fitting using spherical harmonics. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*, pages 257–260. IEEE.