Efficient Acoustic Simulation for Immersive Media and Digital Fabrication

Dingzeyu Li

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Efficient Acoustic Simulation for Immersive Media and Digital Fabrication

Dingzeyu Li

Sound is a crucial part of our life. Well-designed acoustic behaviors can lead to significant improvement in both physical and virtual interactions. In computer graphics, most existing methods focused primarily on improving the accuracy. It remained underexplored on how to develop efficient acoustic simulation algorithms for interactive practical applications. The challenges arise from the dilemma between expensive accurate simulations and fast feedback demanded by intuitive user interaction: traditional physics-based acoustic simulations are computationally expensive; yet, for end users to benefit from the simulations, it is crucial to give prompt feedback during interactions.

In this thesis, I investigate how to develop efficient acoustic simulations for real-world applications such as immersive media and digital fabrication. To address the above-mentioned challenges, I leverage precomputation and optimization to significantly improve the speed while preserving the accuracy of complex acoustic phenomena. This work discusses three efforts along this research direction: First, to ease sound designer's workflow, we developed a fast keypoint-based precomputation algorithm to enable interactive acoustic transfer values in virtual sound simulations. Second, for realistic audio editing in 360° videos, we proposed an inverse material optimization based on fast sound simulation and a hybrid ambisonic audio synthesis that exploits the directional isotropy in spatial audios. Third, we devised a modular approach to efficiently simulate and optimize fabrication-ready acoustic filters, achieving orders of magnitudes speedup while maintaining the simulation accuracy. Through this series of projects, I demonstrate a wide range of applications made possible by efficient acoustic simulations.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Changxi Zheng. He not only shows me how to conduct research, but also how to be a good researcher. Thanks to Changxi's openness in research vision, I have the opportunity to explore both the virtual world in simulation and the real world in fabrication.

I would also like to thank my committee members who gave great suggestions on improving this thesis. Thank you, Prof. Steven Feiner, Prof. Shree Nayar, Prof. David Levin, and Prof. Daniele Panozzo.

I want to thank my great collaborators from both universities and industry. Throughout our collaborations and my internships, I realized that communication is also a crucial part of the research process. It is my honor to work with you. Thank you – Gabriel Cirio, Yun Fei, Eitan Grinspun, Timothy Langlois, David Levin, Wojciech Matusik, Avinash Nair, Shree Nayar, Miguel Otaduy, Yuan Tian, Weiwei Xu, Yin Yang.

I am grateful to my undergrad advisor Prof. Chi-Keung Tang at HKUST for guiding me into the world of vision and graphics research.

My PhD life at Columbia is so joyful and fulfilling with colleagues and friends in New York. It is my pleasure to spend my grad school years with you all: Yinxiao, Jinyu, Henrique, Jie, Raymond, Zichen, Jing, Alec, Gabriel, Yonghao, Angela, Fang, Peter, Chang, Oded, Tim.

If it weren't for all the Computer Science staff, my PhD journey would not be so smooth. Thank you, Anne, Jessica, Elias, Cindy, Corliss, Twinkle, Daisy for taking care of all the logistics.

Last and most importantly, I want to thank my family and Jialei. I thank my parents for fostering my independence and supporting me to pursue better education. I thank Jialei for always standing by my side and growing together into better persons. The PhD journey is much tougher than I initially

anticipated. It is your understanding, encouragement, and love that support me throughout the grind. I made it, because of you.

To mom, dad, and Jialei

Chapter 1

Introduction

Physics-based acoustic simulation in computer graphics has witnessed tremendous progress over the past decades. Parallel to traditional visual renderings of virtual scenes, now we are also able to simulate the sounds with more accurate and efficient algorithms to simulate the excitation mechanism and propagation process. The goal of most existing sound simulation techniques is to add realistic physics-based synchronized audio to existing virtual animation. In the past decade, we have seen a growing number of phenomena that can be reproduced in simulation, for example, thin shell [Chadwick *et al.*, 2009a], fracture [Zheng and James, 2010], fire [Chadwick and James, 2011], cloth [An *et al.*, 2012], and fluids [Langlois *et al.*, 2016].

An emerging area is the application of these simulation algorithms in the presence of real-world recordings. For example, when we compare simulated sounds and recorded ones side by side, there are still discrepancies between them in terms of realism. With the emergence of hardware devices like virtual reality headsets, this difference is further amplified during immersive playbacks. Meanwhile, another challenging scenario for simulation is to predict acoustic behavior using simulation, especially with complex shapes. Although personalized fabrication devices like 3D printers are becoming more accessible, there is no existing tool that supports interactive and accurate simulation

of acoustic behavior for customized complex shapes. My thesis research is to develop realistic sound simulation algorithms that close the gap between virtual and real auditory application.

Different from most previous research that has focused on pure virtual simulation, my thesis explored the use of efficient acoustic simulation techniques to bridge the gap between sound simulations and real-world applications in interactive material selection via efficient precomputation, rapid prototyping of acoustic filters enabled by fast simulation, and immersive spatial audio generation for 360 videos. In my thesis, I develop a suite of algorithms for efficient acoustic simulations that enable a wide range of applications.

In the following, I will outline the structure of this thesis.

Chapter 3 presents an interactive sound simulation framework that supports interactive material parameters editing for rigid body animations [Li *et al.*, 2015]. In practice, material parameters like Young's modulus and Poisson ratio are not exact, since they are measured and tabulated in a range of values. To achieve a desired sound, one usually needs to tune the parameters many times. It is very expensive to compute sound radiation which is modeled by the acoustic transfer values and depends on the input material parameters. In our system, we first select a set of key positions around a vibrating geometry. At every key position, we precompute a frequency-sweeping transfer function with asymptotic Padé approximant expansion and frequency-adaptive mesh simplification during Helmholtz solves. With our proposed precomputation technique, we generate various sound effects without expensive recomputation. At runtime, we construct multipole expansion coefficients from key-position transfer values and evaluate resulting transfer function at any given position interactively.

Chapter 4 looks at the design and optimization of acoustic filters. We propose *Acoustic Voxels*, a computational optimization method to design acoustic filter given an input geometry and high-level frequency requirement [Li *et al.*, 2016]. Our idea is to assemble basic shape primitives into a

complex geometry, one that produces the desired acoustic filtering. We show that these primitives, albeit simple individually, offer a large design space for acoustic filters when *modularly* joined into a complex assembly. This modular scheme also permits fast and accurate estimation of the acoustic performance of a given assembly. For the resulting assembly structure, we combine a stochastic optimization method for the topology of the assembly with a gradient-based quasi-Newton method for computing the geometric parameters of each primitive shape in the assembly. This allows automatically optimizing its structure to achieve target acoustic filtering properties while satisfying geometric constraints of overall shape.

Chapter 5 investigates immersive spatial audio in a new medium, 360° videos [Li *et al.*, 2018]. Unlike visual contents, the creation of spatial audio in immersive audios is challenging. We propose to produce spatial audio by combining a lightweight measurement of room acoustics and a fast geometric acoustic simulation. We first record a single acoustic impulse response (IR) in a room using a readily available mono-channel microphone. We develop an optimization approach that estimates the acoustic material properties associated with the room, based on the measured IR. Then, provided any 360° footage captured in the same environment, our method outputs the 360° video with an accompanying ambisonics spatial soundtrack. The resulting soundfield captures the spatial sound effects at the camera location, even if the camera is dynamic, as if the input audio is emitted from a user-specified sound source in the environment.

Chapter 6 concludes this thesis and discusses several related future directions.

Chapter 2

Related Work

2.1 Rigid-body Acoustic Transfer

The computer graphics community has a long history of synthesizing synchronized sound effects for computer animation [Takala and Hahn, 1992b]. Modal sound models, based on linear modal analysis, have been widely used to generate plausible contact sounds [van den Doel and Pai, 1996] synchronized with physics-based simulation. They are often constructed using recorded sounds [van den Doel *et al.*, 2001; Ren *et al.*, 2013a] or linear modal analysis [Pentland and Williams, 1989; O'Brien *et al.*, 2002]. More recent development has used modal vibration for synthesizing rigid fracture sound [Zheng and James, 2010], deformable sound [Zheng and James, 2011], and fast interactive sound [Raghuvanshi and Lin, 2006; Bonneel *et al.*, 2008; Ren *et al.*, 2010]. But all these methods are closely coupled with vibration frequencies, and none of them enable fast user editing of modal sound parameters with acoustic transfer functions.

The object's geometry can significantly affect modal sound radiation and change the sound's timbre in a spatially varying way, as demonstrated by James et al. [2006a]. Unfortunately, computing

sound radiation for all vibration modes is very expensive. To improve the performance, existing methods [James *et al.*, 2006a; Chadwick *et al.*, 2009b; Zheng and James, 2011] assume fixed modal vibration frequencies, and precompute an efficient representation of acoustic transfer functions. The precomputation can take many hours. Once it is finished, the user can evaluate transfer values at an arbitrary position almost instantaneously. However, whenever the user adjusts vibration frequencies, the entire acoustic transfer representation needs to be recomputed. [Corbett *et al.*, 2007] developed a system to acquire near-field acoustic transfer field from recorded sounds and synthesize spatial sounds interactively. Yet, this approach relied on an automated measuring system, in which the measurement is closely coupled with each object's specific geometry and material. Different from these approaches, our method only relies on precomputation and allows the user to change modal sound parameters at runtime and still enjoys the high quality of sound synthesis.

For fast estimation of sound wave radiation, O'Brien et al. [2001] adopted the *Rayleigh method* which assigns a monopole on each surface element and summed the sound radiation from all monopoles. This is essentially a first-order approximation of sound radiation, neglecting the fact that the shaped structure also scatters and radiates sound. Furthermore, they considered time delays from the monopoles to the listener. In contrast, we solve the Helmholtz radiation equation but ignore the time delays. We also note that the difference of resulting sounds using the Rayleigh method and the Helmholtz solution has been shown in [James *et al.*, 2006a].

On the other hand, when simulating the acoustics of rooms and concert halls, the dimensions of the rooms or obstacles are many times larger than the sound wavelength. As a result, a variety of geometric acoustical methods have been developed [Funkhouser *et al.*, 1999; Tsingos *et al.*, 2001c; Tsingos *et al.*, 2002], analogous to the geometric optics in image rendering. The geometric acoustic methods enjoy fast performance while producing plausible results. However, when the characteristic dimension of objects becomes comparable to the wavelength, as in our problems, the wave diffraction begins to play an import part, necessitating the modeling of sound wave behaviors usu-

ally described by near-field transfer functions. A few previous works have demonstrated the importance of transfer functions [James *et al.*, 2006a; Corbett *et al.*, 2007] for typical rigid object sounds in computer animations.

There exist several example-based methods on estimating sound parameters from input audio clips. Early work by Pai et al. [2001] estimated rigid modal sound models from recorded sounds and measurements. More recently, Lloyd et al. [2011] analyzed the short-time Fourier Transform and identify the strongest peaks in the spectrogram to estimate modal parameters. Ren et al. [2013a] computed a set of features from given examples of audio clips, and used them to optimize modal sound parameters. While they can produce high-quality parameter estimation, they are not focused on acoustic transfer functions of vibration modes. With our model, we are able to explore modal sound parameter space straightforwardly (see §3.6) and synthesize resulting sounds that take into account the sound radiation effects.

The frequency-varying sound radiation problem has been studied in many engineering applications. Fast frequency sweep methods are most closely related to our method [Pillage and Rohrer, 1990; Lenzi *et al.*, 2013]. The basic idea is to compute Helmholtz solutions at a few key frequencies and interpolate / extrapolate Helmholtz radiation at intermediate points. However, these methods aim to produce engineering accuracy rather than high performance at runtime. Although they provide approximations of frequency-varying Helmholtz solutions, it is nontrivial to evaluate transfer values at an arbitrary point at runtime. In contrast, we aim for fast evaluation of acoustic transfer at any spatial and frequency point. To this end, we propose to use a Prony series representation [Hauer *et al.*, 1990] constructed using adaptive frequency sweeping.

In addition to the sound synthesis from physically based simulation, there exist numerous audio processing software. Almost all these tools rely on signal processing algorithms to edit sound effects such as frequency modulation [Chowning, 1973], reverberation [Smith, 1985], and spectrum adjustment [Strawn, 1987], or use stochastic sound models and granular synthesis methods to pro-

duce natural sound textures [Cook, 2002]. However, these methods lack *automatic synchronization* with computer-simulated animation, and often need to store a large database of sound effects. Our model is complementary to those tools, enabling automatic audiovisual synchronization using physical simulation.

Our method utilizes frequency-adaptive mesh simplification to accelerate the individual Helmholtz solves. There are numerous methods for surface mesh simplification (see a survey by Luebke [2001]). Among them, our method is based on the edge collapse algorithms [Garland and Heckbert, 1997], which coarsen a mesh through a sequence of edge collapse operations. In particular, we augment the method introduced in [Hoppe, 1999] and [Lindstrom and Turk, 1998] to preserve mesh volume as well as volume vibration velocity. The latter is an important quantity to preserve sound radiation power. Consequently, the optimization problem for edge collapse becomes significantly harder: rather than solving a linear system, we need to solve a quadratically constrained quadratic programming (QCQP) problem, for which we propose a staggered iterative algorithm.

The idea of using geometric simplification for efficient acoustic computation has been used in the research of room acoustical modeling. The input CAD models are often simplified architectural models in an exchange for faster computation. Siltanen et al. [2008] proposed a geometry reduction method based on volumetric reconstruction using a modified Marching Cubes algorithm. Further, geometrical acoustical simulation has adopted level-of-detail approaches to adaptively select polygon meshes used in the computation [Tsingos *et al.*, 2007; Pelzer and Vorländer, 2010]. The adaptivity of these approaches is based on incident sound waves for sound auralization. While these approaches mostly focus on room acoustics, our method is concerned with sound radiation produced by the modal vibration of an object. Therefore, the adaptivity of our method is based on the modal vibration frequencies of the object.

2.2 Acoustic Filter Design

Sound simulation The computer graphics community has a long history of simulating sound propagation in a virtual environment [Stettner and Greenberg, 1989; Takala and Hahn, 1992a], starting from the geometric acoustical methods [Funkhouser *et al.*, 1998; Funkhouser *et al.*, 1999; Tsingos *et al.*, 2001a] which are fast but less accurate at low frequencies, and evolving to the wave-based methods [James *et al.*, 2006b; Raghuvanshi *et al.*, 2010a; Mehra *et al.*, 2013a; Raghuvanshi and Snyder, 2014] to improve sound quality. The goal of these work is to add realistic wave scattering and room acoustic effects. These approaches have proven successful in many virtual environment applications, but not for fabricating acoustic structures. Moreover, the geometric scale in those simulations is typically meters or tens of meters, whereas we are interested in the sound propagation in small cavities at the centimeter scale.

Recently, Allen and Raghuvanshi [2015] proposed an interactive method for simulating wave propagation in wind instruments, modeled in 2D. This method produces realistic sound effects in real-time, but is unclear how to apply it for solving our inverse problem. Aside from requiring a physics-based 3D simulation and high accuracy for predicting fabricated results, our method needs a well-defined relationship between the sound transmission and the boundary geometry to formulate a tractable inverse problem.

Acoustic inverse problem Acoustic inverse problems have intrigued scientists for decades, starting from Kac's famous question: "can one hear the shape of a drum?"[Kac, 1966]. While Kac's question is about the vibrational patterns of a shape, similar questions that infer shapes from sound propagation and scattering patterns have been actively studied [Angell *et al.*, 1997; Feijóo *et al.*, 2004]. Monks et al. [2000] optimized room acoustics motivated by the applications in architectural design. Recently, Dokmanić et al. [2013] showed an algorithm for computing a convex polyhedral room shape using acoustic response recorded at multiple microphones. We also address an

acoustic inverse problem, but from a different perspective. Our input is the acoustic response (i.e., impedance or transmission loss) measured at a pair of locations (i.e., between the inlet and the outlet), and our goal is not to reconstruct existing shapes but to construct new structures.

Transmission Line Matrix Based on Huygens' model of wave propagation and the analogy between wave propagation and transmission lines, the Transmission Line Method has been widely used for computing electromagnetic waves [Caloz and Itoh, 2005; Christopoulos, 2006] and acoustics [Munjal, 2014]. It first discretizes the computational domain into interconnected nodes. On the connecting interface, field information is propagated and coupled between adjacent nodes. By breaking down the whole domain into basic nodes, the computational performance can be significantly improved. Our method shares the similar idea, but a key difference lies in the new optimization framework. Our method optimizes the configuration of the nodes both geometrically and topologically, aiming to realize the desired acoustic filtering properties.

Muffler design Noise attenuation is an important topic in many engineering fields. There has been a well established theory for modeling noise reduction in a cavity structure [Ingard, 2009; Munjal, 2014], and numerous approaches for improving a standard muffler have been developed with sub-chamber structures [Selamet *et al.*, 2003], varying inlet and outlet sizes [De Lima *et al.*, 2011], or perforated liners [Chiu, 2010; Munjal, 2014]. However, the optimization for desired target performance is not straightforward. Traditionally, mufflers are often analyzed using finite element methods and then used in a sensitivity analysis to compute the derivatives of the muffler metric with respect to shape parameters. In general, this is a computationally expensive process.

The application of our method for muffler design takes a different approach, namely tiling simple resonator shapes, without choosing a specific parametric shape *a priori*. Meanwhile, precomputed filtering properties of primitive resonators sidestep the expensive finite-element solves during the

optimization and thus allows us to optimize for complex structures.

Computational design of music instruments Our method can be applied to customize wind instruments, although that is not a primary goal of our work. Related to this aspect, existing work has explored the optimization of the bore shapes for brasswind instruments [Kausel, 2001; Noreland *et al.*, 2010]. Similar to our optimization target, Braden et al. [2009] use the input impedance of the instrument in the objective function to optimize bore shapes. These methods typically focus on a specific family of shapes and thereby formulate a continuous optimization problem. Our method, in contrast, aims to create acoustic filters using an arbitrary shape for a range of applications beyond wind instruments. More recently, Zoran [2011] has demonstrated the use of 3D printers for creating plausible wood instruments and for exploring new designs without any numerical optimization.

In computer graphics, Umetani et al. [2011] have develop the first interactive tool for designing metallophones. The tool aims for interactivity but not for solving the inverse problem. Recently, Bharaj et al. [2015] have explored the inverse computational design of metallophones and have proposed a stochastic optimization method for this purpose. Unlike ours, both approaches focus on the modal vibrational sounds from solid vibrations but not the sound propagation inside a chamber.

Microstructure design Recently in computer graphics, there has been a variety of work on designing macroscopic mechanical material properties through controlling their microscopic structures, based on the inverse homogenization theory [Sigmund, 1994]. Along this line of research, existing work has used a data-driven approach to control nonlinear elasticity [Bickel *et al.*, 2010] with multi-material 3D printing, while others tile precomputed structural patterns [Panetta *et al.*, 2015; Schumacher *et al.*, 2015] to obtain user-specified elastic properties. While we also combine small-scale primitives, in order to affect sound waves, the geometric size of our primitives is of a few

centimeters, much larger than the microstructure scales in these approaches. Furthermore, rather than the elastic behaviors of microstructures, we focus on the sound propagation through the primitives.

Acoustic in HCI Recent development in passive acoustic sensing inspired new HCI applications, such as the recent tangible input devices by analyzing the sound produced by a comb-like structure [Savage *et al.*, 2015]. More relevant to our method, Laput et al. [2015] proposed Acoustruments to recover information from audio signals recorded through ducts. None of these previous works considers the inverse problem of acoustic optimization. Our method complements to those work and offers a computational tool to develop new HCI applications, as we will demonstrate in §4.5.4.

Contributions Compared to previous work, our method has the following contributions: (i) We propose to construct acoustic filters using primitive resonators. (ii) With modular assemblies, we develop a numerical optimization method to construct desired acoustic filters while sidestepping expensive finite-element solves. (iii) We demonstrate the use of our primitives and optimization method in the context of different applications including a new application that embeds acoustic signatures into 3D printed objects.

2.3 Spatial Audio in 360° Videos

Recent advances in 360° video research have focused mostly on improving *visual* quality. Rich360 and Jump designed practical camera systems and developed seamless stitching with minimal distortion, even for high-resolution 360° videos [Lee *et al.*, 2016; Anderson *et al.*, 2016]. To capture stereo omni-directional videos, Matzen et al. [2017] built a novel capturing setup from off-the-shelf

components, providing a more immersive viewing experience in head-mounted displays with depth cues. Kopf [2016] introduced a 360° video stabilization algorithm for smooth playback in the presence of camera shaking and shutter distortion. Our work improves the audio experience in existing 360° videos, working in tandem with existing methods for capturing, post-processing, and playback for immersive visual media.

Spatial audio in virtual reality (VR) is also crucial to provide convincing immersion. Most recent work aims to enable efficient rendering of spatial audio at real-time rates. Schissler et al. [2016] proposed a novel analytical formulation for large area and volumetric sound sources in outdoor environments. Constructing spatial room impulse responses (SRIR) with geometric acoustics is expensive due to the number of rays and the disparity in energy distribution. Schissler et al. [2017b] partition the traditional impulse response (IR) into segments and project each segment onto a minimal order spherical harmonics bases to retain the perceptual quality. We build upon the concept of SRIR and observe that late reverberation is diffuse, which means that the late IR tail is uniform not only spatially but directionally. Our method combines early IR simulation with estimated material parameters and recorded late IR tails to generate scene-aware audio for 360° videos.

Recording and reproducing spatial audio provides the fundamental building blocks for more advanced virtual auditory manipulation algorithms. In the seminal work by Li and Duraiswami [2006], a hemispherical microphone array was used to record the spatial soundfield. Using spherical beamforming, the authors demonstrated 3D soundfield reproduction in headphone-based scenarios. Later a vision-based system combined cameras with microphones to enable immersive joint audiovisual sensing [O'Donovan *et al.*, 2007]. To efficiently incorporate room acoustics, Zotkin et al. [2004] computed early reflections and reused the late tail regardless of the microphone/speaker locations. Inspired by the hybrid idea, we introduce an optimized simulation for the early part. To overcome the inherent limitation of geometric acoustic simulators, we also propose a frequency modulation algorithm to compensate for the wave-based room resonance effects.

The simulation of sound propagation has been widely studied [Vorländer, 2008; Bilbao, 2009]. Wave-based methods usually provide high accuracy with expensive computation [Raghuvanshi *et al.*, 2009]. Alternatively, geometric acoustic (GA) methods can be used, which make the high-frequency Eikonal ray approximation [Savioja and Svensson, 2015]. These methods often bundle rays together and trace as beams for efficiency [Funkhouser *et al.*, 1998]. While traditional GA does not include diffraction effects, they can be approximated via the uniform theory of diffraction for edges that are much larger than the wavelength [Tsingos *et al.*, 2001b; Schissler *et al.*, 2014]. We use the GA method proposed by Cao et al. [2016], which exploits bidirectional path tracing and temporal coherence to provide significant speedups over previous work. For fast auralization in VR, many methods precompute IRs or wavefields [Pope *et al.*, 1999; Tsingos, 2009; Raghuvanshi and Snyder, 2014]. Raghuvanshi et al. [2010a] precompute and store one LRIR per room, similar to our method. We show how to use recorded IRs to optimize acoustics materials for simulation, and also how to directly use the recorded IR tails instead of simulating them, reducing the computation time and memory requirements. Moreover, our method accounts for a particular wave effect, the room resonances, using a frequency modulation algorithm, which further improves the generated audio quality.

To synthesize scene-aware audio, optimal material parameters are needed in the simulation. Given recorded IRs, we estimate the material parameters that best resemble the actual recording. For rigid-body modal sounds, Ren et al. [2013b] optimized the material parameters based on recordings and demonstrated the effectiveness of optimized parameters to virtual objects. Most related to ours is Schissler et al. [2017a] where a pretrained neural network is used to classify the objects, followed by an iterative optimization process. Every iteration requires registering the simulated IR with a measured IR and solving a least-squares problem. We draw inspirations from inverse image rendering problems [Marschner and Greenberg, 1998], and derive an analytical gradient to the inverse material optimization problem, which we solve in a nonlinear least-squares sense. Our optimization

runs in seconds, tens of times faster than previous work.

While our method aims to ease the audio editing process, this is a broad area with an abundance of prior work. Most methods strive to provide higher-level abstractions and editing powers, to help users avoid non-intuitive direct waveform editing. VoCo [Jin *et al.*, 2017] allows realistic text-based insertion and replacement of audio narration using a learning-based text to speech conversion which matches the rest of the narration. Germain et al. [2016] present a method for equalization matching of speech recordings, to make recordings sound as if they were recorded in the same room, even if they weren't. Rubin et al. [2013] present an interface for editing audio stories like interviews and speeches, which includes transcript-based speech editing, music browsing, and music retargeting. Like previous work, we aim to match the timbre of generated sounds with that of recordings. Moreover, we are able to produce spatial audio that blends in seamlessly with existing 360° videos, and provide a high-level "geometric" effect which can be applied to audio.

Chapter 3

Interactive Rigid-body Acoustic Transfer

3.1 Introduction

Modal sounds are widely used for synthesizing plausible solid-object sounds synchronized with computer-simulated animations (e.g., see [van den Doel *et al.*, 2001; O'Brien *et al.*, 2002; Zheng and James, 2011; Ren *et al.*, 2013a]). The standard pipeline consists of two steps: (i) integration of surface vibrations followed by (ii) the computation of sound radiation. The former produces surface motions that are driven by external forces and vibrate at individual modal frequencies, while the latter accounts for wave phenomena such as diffraction and interference that can recognizably change the sound's timbre [James *et al.*, 2006a]. Both steps are closely coupled with modal vibration frequencies.

The most expensive step of generating a modal sound is computing sound radiation. For every vibration mode with a frequency ω, it can be computed by solving a frequency-domain wave equation, the *Helmholtz equation*,

$$\nabla^2 p(\boldsymbol{x}) + k^2 p(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega, \tag{3.1}$$

where $p(\boldsymbol{x})$ is the acoustic transfer value at $\boldsymbol{x}$, $k = \omega/c$ is the wavenumber of the corresponding vibration mode, and c is the speed of sound. To accelerate this step, various methods have been devised [Ciscowski and Brebbia, 1991]. In computer graphics, James et al. [2006a] introduced precomputed acoustic transfer (PAT), wherein, after hours of precomputation of equivalent point sources, fast runtime transfer evaluation is achieved at any listening location. However, in all of these methods, the entire Helmholtz solution needs to be recomputed whenever the frequency ω is changed.

The tight dependence of modal sound radiation on its vibration frequencies as well as its expensive (pre-)computation of sound radiation give rise to many difficulties when one starts to tweak model sound parameters for desirable sound effects. In practice, tuning parameters is almost unavoidable, as the material parameters (e.g., the Young's modulus) are measured and tabulated in a range of values, and there are no generally accepted damping values [Adhikari and Woodhouse, 2001]. Both kinds of parameters directly affect modal frequencies (see §3.2), which have been found critical for achieving desired sound characteristics [Klatzky *et al.*, 2000]. Unfortunately, when the user changes these parameters and thus the frequencies, it becomes necessary to recompute the entire modal sound, leading to a rather inefficient parameter tuning cycle.

In light of this, we propose a new method that *decouples* surface modal vibration and acoustic transfer evaluation from modal vibration frequencies. It allows the user to freely change modal frequencies at runtime, and quickly synthesize resulting sounds at an arbitrary listening location.

At first glance, one simple approach to enable runtime editing of modal frequencies is to precompute individual modal sound models using a set of frequency samples, and rely on runtime interpolation to approximate with user-specified vibration frequencies. Unfortunately, it is unclear how we should interpolate between different models of modal sound. Moreover, the acoustic transfer $p(\boldsymbol{x})$ is highly oscillatory with respect to the vibration frequency (see §3.4 and Figure 3.5). As a result, such an approach will need lots of frequency samples for plausible runtime interpolation, causing a

prohibitively long precomputation time and an overwhelming memory footprint.

In our approach, we first select a set of key positions around a vibrating object. At every key position we precompute a frequency-sweeping transfer function compactly represented using Prony series. We accelerate the precomputation by devising two key techniques: (i) we solve Helmholtz equations only at a few carefully selected frequency samples. And for each frequency sample, we build an asymptotic Padé approximant in frequency domain to evaluate transfer values at nearby frequencies. (ii) To speed up Helmholtz solves, we propose a frequency-adaptive mesh simplification algorithm; for low-frequency boundary element (BEM) Helmholtz solves, we simplify the mesh more aggressively in exchange for larger computational speedup. At runtime, given a user-specified modal frequency, we represent the resulting transfer function at any spatial position using an acoustic multipole expansion. We first evaluate key-position transfer values, which are in turn used to construct a small least-squares problem to estimate the multipole expansion coefficients.

With our proposed precomputation technique, we are able to generate various sound effects without expensive recomputation. This greatly eases the parameter tuning for different sound characteristics, whether one desires high-pitch long-ringing metal sounds or low-tone quickly damped wood-like sounds. We can explore the parameter space and quickly hear the sound feedback (Figure 3.1). The resulting sounds are almost identical to the ones using expensive full recomputation, both qualitatively and numerically.

Our technique also enables more control of sound characteristics for animators wishing to add synchronized modal sounds. We explore examples of nonlinear time-varying modal sound effects using user-guided nonphysical change of frequencies.

3.2 Modal Sound Preliminary

Before presenting the details of our method, we briefly review the widely used modal sound model (see [Shabana, 1991; O'Brien *et al.*, 2002; James *et al.*, 2006a] for details) and clarify the frequency-related parameters that can be freely changed in our model.

Modal Vibration First, a solid object vibration is approximated by a linear vibration equation,

$$\mathsf{M}\ddot{\boldsymbol{u}} + \mathsf{D}\dot{\boldsymbol{u}} + \mathsf{K}\boldsymbol{u} = \boldsymbol{f}_{ext}, \tag{3.2}$$

where M, K, and D are respectively the mass, stiffness, and damping matrices depending on the object materials, $\boldsymbol{u} \in \mathbb{R}^{3n}$ describes the finite element nodal displacement with n nodes, and $\boldsymbol{f}_{ext} \in \mathbb{R}^{3n}$ is the external force driving the vibration. The damping matrix D is usually approximated using the *Rayleigh damping* model [Shabana, 1991], i.e., $\mathsf{D} = \alpha\mathsf{M} + \beta\mathsf{K}$, where the scalars α and β are user-specified parameters. Linear modal analysis then solves a generalized eigenvalue problem $\mathsf{KU} = \mathsf{MUS}$ to compute a modal shape matrix U and a diagonal eigenvalue matrix S. The former describes the vibration pattern of each mode while the latter indicates the square of undamped natural frequencies, i.e., $\mathsf{S}_{i,i} = \omega_i^2$. Substituting $\boldsymbol{u} = \mathsf{U}\boldsymbol{q}$ and then premultiplying U on both sides of (3.2) decouples the system into a set of 1D second-order ordinary differential equations (ODEs), each of which is an ODE describing the modal vibration of a single mode i, namely,

$$\ddot{\boldsymbol{q}}_i + d_i\dot{\boldsymbol{q}}_i + \omega_i^2\boldsymbol{q}_i = \mathsf{U}_i^T\boldsymbol{f}_{ext}, \tag{3.3}$$

where d_i is the damping parameter of mode i, and U_i is the i-th column of U.

Sound Radiation A vibration mode with an observed frequency ω produces propagating sound waves that have a wavenumber $k = \omega/c$. A standard tool to model its sound radiation is the

Helmholtz equation (3.1), which is coupled with surface modal vibration through a Neumann boundary condition defined on the object surface S,

$$\frac{\partial p}{\partial \boldsymbol{n}} = -i\omega\rho\boldsymbol{v} \quad \text{on } S, \tag{3.4}$$

where i is the imaginary unit, ρ is the air density, and $\boldsymbol{v}$ is the mode's time-harmonic vibration velocity along the surface normal direction, computed as $\boldsymbol{v} = i\omega(\boldsymbol{n} \cdot \tilde{\boldsymbol{u}}_i)$, where $\boldsymbol{n} \cdot \tilde{\boldsymbol{u}}_i$ is the normal-direction modal displacement of a mode i. Note we also use i to represent mode index when there is no ambiguity. Solving (3.1) for each mode i results in a complex-valued transfer function $p_i(\boldsymbol{x})$. Finally, following the approximation in [James *et al.*, 2006a], the sound wave at a listening position $\boldsymbol{x}$ is computed as a weighted summation of all audible vibration modes,

$$s(\boldsymbol{x}) = \sum_i |p_i(\boldsymbol{x})| \boldsymbol{q}_i(t). \tag{3.5}$$

This expression is accurate up to a phase, ignoring the time delay of sound propagation. The Helmholtz solution describes modal sound radiation affected by the object's own geometry, and ignores environment acoustics. This is sufficient in our problem because environment acoustics are independent from a modal sound model; if the environment is known, one can easily feed the resulting sound of our model to any sound auralization methods (e.g., [Tsingos *et al.*, 2001c; Raghuvanshi *et al.*, 2010b; Mehra *et al.*, 2013b]) to add room acoustic effects.

Frequency-Related Parameters As observed by Klatzky et al. [2000], two sets of parameters are of particular importance for achieving desired sound characteristics: *vibration frequencies*, ω_i, that determine sound pitch, and *damping coefficients*, d_i, that affect the timbre of particular materials. For instance, a small damping value results in the long ringing sounds that metal or porcelain objects often produce, whereas a large damping value tends to produce sounds more like wood or stone. We

note that the damping coefficients d_i are also frequency-related; they affect the observed *damped natural frequency* through the relationship $\tilde{\omega}_i = \sqrt{\omega_i^2 - d_i^2/4}$. This frequency value is used for computing the wavenumber k, and thus affects the Helmholtz solution. In standard modal sound models, the user can change material parameters such as Young's modulus to adjust ω_i, and change α and β values in Rayleigh damping to control d_i. In our implementation, we change the scales of Young's modulus, which we call stiffness scales, and damping scales. They form a 2D parameter space (see Figure 3.1), in which the modal shape matrix U remains constant. We also explore the examples that allow the user to change ω_i and d_i directly and individually (see §3.6).

3.3 Interactive Sound Synthesis Algorithm

In this section, we introduce our runtime sound synthesis algorithm while deferring the precomputation details until §3.4. At runtime, we take as input an animation sequence, the user-specified ω_i and d_i, and the contact forces that appear on the right-hand side of (3.2) to drive the surface vibration. Given a listening location $\boldsymbol{x}$, it computes surface modal vibration $\boldsymbol{q}_i$ for every mode (in §3.3.1), and evaluates the transfer function $p_i(\boldsymbol{x})$ (in §3.3.2). The final sound is computed using the superposition (3.5) of individual modes. For simplicity of presentation, we describe the sound synthesis algorithm for a single mode. An outline of our runtime algorithm is shown in Algorithm 1.

3.3.1 Vibration Integration

We first solve the decoupled 1D modal vibration equation (3.3), where the external force $\boldsymbol{f}_{ext}$ is the contact forces resulting from the simulated animation. Many previous methods [Hamming, 1983; James and Pai, 2002] solve this inhomogeneous second-order ODE using a digital Infinite Impulse Response (IIR) filter. Our implementation uses the fourth-order Runge-Kutta method [Press *et al.*,

Algorithm 1 Runtime Sound Computation for Mode i

Require: frequency ω_i of mode i and listening position $\boldsymbol{x}$

1: **procedure** SOUNDEVAL($i, \boldsymbol{x}$)
2: Compute $q_i(t)$ at audio rate by integrating (3.3) ▷ §3.3.1
3: Compute transfer $p(\omega_i)$ at key positions using (3.10) ▷ §3.4.1
4: Estimate M_n^m using a least-squares solve (3.8) ▷ §3.3.2
5: Compute transfer $p(\boldsymbol{x})$ using M_n^m and (3.6)
6: Compute $\sum_i |p(\boldsymbol{x})| q_i(t)$ at the audio rate
7: **end procedure**

2007], since we found it has comparable performance but higher accuracy than the digital IIR filter, especially when the user specifies time-varying ω_i and d_i values, as in the examples of §3.6 (see Figure 3.2).

3.3.2 Transfer Estimation via Least-Squares

A main challenge for runtime sound synthesis is the evaluation of transfer values $p_i(\boldsymbol{x})$. This is because p_i, the solution of the Helmholtz equation (3.1), is frequency-dependent. Whenever the user changes frequency parameters, we need to update $p_i(\boldsymbol{x})$, but solving the Helmholtz equation from scratch is impractical and computationally very expensive.

Multipole Approximation To allow the user to freely change the listening location $\boldsymbol{x}$ while editing a sound, we need a compact representation of $p(\boldsymbol{x})$ in the spatial domain. Similar to [Zheng and James, 2010], we represent a Helmholtz solution $p(\boldsymbol{x})$ using a single point multipole expan-

sion [Gumerov and Duraiswami, 2004], which takes an expansion form,

$$p_i(\boldsymbol{x}) \approx ik \sum_{n=0}^{N} \sum_{m=-n}^{n} S_n^m(\boldsymbol{x}, \bar{\boldsymbol{x}}_0) M_n^m(\omega). \tag{3.6}$$

Here $\bar{\boldsymbol{x}}_0$ is the expansion center near the object. In practice, we always place it at the object's center of mass. We follow the rule of thumb [Liu, 2009; Zheng and James, 2010] and set the expansion order $N = \max(\frac{1}{4}kL, 4)$ ($N \leq 18$ in all our examples). S_n^m are singular Helmholtz basis functions, $S_n^m(\boldsymbol{r}) = h_n^{(2)}(kr)Y_n^m(\theta, \phi)$, where $\boldsymbol{r} = (r, \theta, \phi)$ is the spherical coordinate of $\boldsymbol{x} - \boldsymbol{x}_0$, $h_n^{(2)} \in \mathbb{C}$ are spherical Hankel functions of the second kind, and $Y_n^m \in \mathbb{C}$ are spherical harmonics. The expansion coefficients M_n^m depend on the modal vibration frequency ω, and are what we need to quickly update when ω is changed at runtime.

Previous methods of computing M_n^m (e.g. [Gumerov and Duraiswami, 2004; Zheng and James, 2010]) integrate the results of a boundary element (BE) solve of (3.1) over the entire object surface. Both the BE solve and surface integral are expensive. One might precompute a set of M_n^m using frequency values sampled in a frequency range, and use the interpolated M_n^m at runtime. However, as shown in Figure 3.3, M_n^m at high orders fluctuates dramatically as the frequency value sweeps. Consequently, the frequency needs to be densely sampled to interpolate M_n^m, leading to a prohibitively long precomputation time.

Fast Least-Squares Approximation of M_n^m Inspired by the subspace construction for shape deformations [Meyer and Anderson, 2007], we approximate M_n^m using a small-scale least-squares approximation. Here, the Helmholtz basis functions S_n^m construct a set of reduced-space bases of the Helmholtz solution. We estimate the basis coefficients M_n^m based on the transfer values at a set of key positions, and use the resulting M_n^m to compute transfer value $p_i(\boldsymbol{x})$ at any listening location $\boldsymbol{x}$.

First, we sample a set of *key positions* $\boldsymbol{x}_j, j = 1, ..., J$ outside of the object (see Figure 3.4). In the precomputation, we construct a frequency-sweeping transfer function $p_j(\omega)$ at $\boldsymbol{x}_j$. The representation of $p_j(\omega)$ is compact, adding very little memory overhead over the standard modal sound model (see §3.5). And its construction requires only coarse frequency samples. We defer the details of its construction until §3.4.1. Here we use the precomputed representation to evaluate $p_j(\omega)$ with user-specified ω at every key position $\boldsymbol{x}_j$ and stack them into a vector $\tilde{\boldsymbol{p}}$.

The summation of (3.6) can be expressed in matrix form, $\tilde{\boldsymbol{p}} = \mathsf{A}\boldsymbol{m}$, where $\boldsymbol{m}$ stacks all the coefficients M_n^m, and A consists of the Helmholtz basis function values at all selected positions $\boldsymbol{x}_j$. Concretely, they have the form

$$\mathsf{A} = \begin{bmatrix} S_0^0(\boldsymbol{x}_1, \bar{\boldsymbol{x}}_0) & \dots & S_N^N(\boldsymbol{x}_1, \bar{\boldsymbol{x}}_0) \\ \vdots & \ddots & \vdots \\ S_0^0(\boldsymbol{x}_J, \bar{\boldsymbol{x}}_0) & \dots & S_N^N(\boldsymbol{x}_J, \bar{\boldsymbol{x}}_0) \end{bmatrix} \text{ and } \boldsymbol{m} = \begin{bmatrix} M_0^0 \\ \vdots \\ M_N^N \end{bmatrix}. \tag{3.7}$$

We then estimate the unknown coefficients M_n^m by solving the least squares problem

$$\mathsf{A}\boldsymbol{m} = \tilde{\boldsymbol{p}}. \tag{3.8}$$

As long as the number of key positions is larger than the number of columns of A, we have an over-constrained and complex-valued least-squares problem, and thus the solution is unique. Recall that the order of multipole expansion (3.6) is small (i.e., $N \leq 18$). The number of columns is also small (i.e., $N^2 \leq 324$), and thus the least-squares problem can be efficiently solved at runtime. Once M_n^m are computed, we substitute them into (3.6) to evaluate the transfer value at any location $\boldsymbol{x}$. In §3.5, we validate the accuracy and convergence of our transfer evaluation algorithm.

When sampling key positions, we need to cover the region where the listener will be located. We therefore select three spheres centered at the object's center of mass $\bar{\boldsymbol{x}}_0$ with radii of 1.6, 2.6, and

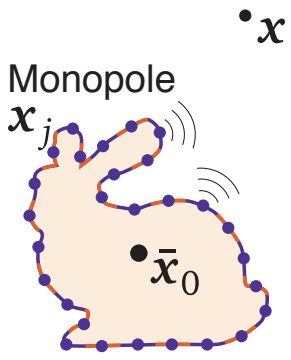
3.4 times the object's geometric size. We then uniformly sample positions over the spheres (see Figure 3.4). In §3.5, we validate the numerical accuracy and convergence of this scheme.

3.4 Precomputation of Helmholtz Equations

The core goal of our precomputation is to construct a representation of frequency-sweeping transfer function $p_j(\omega)$ for every key position $\boldsymbol{x}_j, j = 1, ..., J$ and every vibration mode. This representation is used at runtime to construct the right-hand-side vector $\tilde{\boldsymbol{p}}$ in (3.8). For a mode i with a natural vibration frequency ω_0, we allow the user to adjust its vibration frequency in the range $\mathcal{R} = [\omega_0 - \Delta\omega, \omega_0 + \Delta\omega]$. In practice, we allow the runtime frequency adjustment in a range of 5kHz (i.e., $\Delta\omega = 2.5\text{kHz} \cdot 2\pi$).

A simple approach is to compute $p_j(\omega_t)$ at a set of frequency samples $\omega_t, t = 1, \ldots, T$ in $\mathcal{R}$, and interpolate to obtain $p_j(\omega)$. However, this approach requires a large number of frequency samples, since $p_j(\omega)$ oscillates at a high frequency as shown in Figure 3.5. And evaluation of transfer samples $p_j(\omega_t)$ at different frequencies requires expensive individual Helmholtz solves. Therefore, we seek to sample $\mathcal{R}$ using a sparse set of ω_t. Our algorithm strives to (i) avoid as many Helmholtz solves as possible (in §3.4.2) and (ii) improve the solving performance (in §3.4.3). We start by first presenting an efficient representation of $p_j(\omega)$ using sparse transfer samples $p_j(\omega_t)$. For simplicity, we will drop the subscript j, because we will consider transfer values at a single key position $\boldsymbol{x}_j$.

3.4.1 Frequency-Sweeping Transfer Representation



To reveal why $p(\omega)$ is oscillatory, consider a first-order approximation of sound radiation. As commonly used in the *Rayleigh method* [Cremer *et al.*, 2005], we can discretize a vibrating surface into small elements,

and estimate its radiation (up to first order) by placing a monopole on every element and summing up their contributions. Namely, the acoustic transfer at $\boldsymbol{x}$ is estimated as

$$p(\boldsymbol{x}) \approx \sum_{j=1}^{N} C_j e^{-ikr_j} = \sum_{j=1}^{N} C_j e^{-i\frac{\omega}{c}r_j}, \tag{3.9}$$

where C_j is the weight of a monopole on element j and r_j is the distance from $\boldsymbol{x}$ to the position of j-th monopole (i.e., $r_j = |\boldsymbol{x} - \boldsymbol{x}_j|$). This expression clearly shows that as ω sweeps in a range $\mathcal{R}$, $p(\boldsymbol{x})$ oscillates because $e^{-i\frac{\omega}{c}r_j}$ is harmonic with respect to ω. In fact, for a far-field listener, r_j is large, and thus $p(\boldsymbol{x})$ oscillates strongly.

Interpolation of $p(\omega)$ We propose the following scheme to interpolate $p(\omega)$. For every acoustic transfer sample $p(\omega_t)$, we compute $\tilde{p}(\omega_t) = e^{ik_t r} p(\omega_t)$, where k_t is the wavenumber ω_t/c, and $r = |\boldsymbol{x}_j - \bar{\boldsymbol{x}}_0|$ is the distance from a key position $\boldsymbol{x}_j$ to the object's center of mass. We claim that $\tilde{p}(\omega)$ is much smoother than $p(\omega)$ (see Figure 3.5), and thus we can easily construct an interpolation of $\tilde{p}(\omega_t)$ in $\mathcal{R}$. To understand the reason, we again look to the approximation (3.9), and have $e^{ikr}p(\omega) \approx \sum_j C_j e^{-ik(r_j - r)}$. For a far-field listening location, we have $|r_j - r| \ll |r|$. Therefore, $e^{-ik(r_j-r)}$ and hence $e^{ikr}p(\omega)$ are much less oscillatory. Lastly, given a frequency ω, we interpolate $\tilde{p}(\omega)$ and compute the transfer value using $p(\omega) = e^{-ikr}\tilde{p}(\omega)$.

Representation of $\tilde{p}(\omega_t)$ using Prony's Method The remaining question is how to represent $\tilde{p}(\omega_t)$. Since we will create a representation of $\tilde{p}(\omega)$ for every key position and every vibration mode, the representation needs to be compact. Note $\tilde{p}(\omega)$ still oscillates with respect to ω, albeit smoothly, because of the term $e^{ik(r_j-r)}$. This suggests that we should use a small number of harmonic basis functions to interpolate $\tilde{p}(\omega_t)$. Fourier basis functions were considered; they are efficient for representing periodic signals. However, in our case, the amplitude of $p(\omega_t)$ generally tends to decrease

as ω increases, because the high-frequency waves dissipate energy faster than low-frequency waves do. These damped signals require more Fourier bases for a plausible representation (see Figure A.1 in Section A.6). Instead, we propose to use *Prony's method* [Hauer *et al.*, 1990; Lobos *et al.*, 2003], which approximates a uniformly sampled signal using a series of weighted complex exponentials. In our case, it approximates $\tilde{p}(\omega)$ as

$$\tilde{p}(\omega) \approx \sum_{i=1}^{N} c_i e^{\mu_i \omega} \tag{3.10}$$

where c_i and μ_i are complex values determined using the transfer samples $\tilde{p}(\omega_t)$. For readers not familiar with this method, we list the details in Section A.1. Here we highlight its advantages in our problems. (i) Prony series has been known for its efficiency on estimating damping coefficients apart from frequency, phase, and amplitude [Lobos *et al.*, 2003]. It requires only sparse samples to represent the signals (i.e., $2N$ samples for N harmonic components). (ii) It offers a compact representation of $\tilde{p}(\omega)$, allowing fast runtime evaluation of $p(\omega)$. With precomputed c_i and μ_i, we use the Prony series (3.10) to construct the right-hand-side vector in the least-squares problem (3.8) for estimating the multipole coefficients M_n^m as described in § 3.3.2. Figure 3.5 shows that $N = 6$ is sufficient for a close approximation in our experiments. Table 3.1 lists the storage needed for runtime use, less than 25MB per model. (iii) Computing the parameters c_i and μ_i is fast, involving only two small least-squares solves and a polynomial root-finding (see Section A.1).

3.4.2 Adaptive Frequency Sweep

Creating a Prony's representation of $\tilde{p}(\omega)$ takes as input a set of transfer values $p(\omega_t)$ at uniformly sampled $\omega_t \in \mathcal{R}, t = 1, .., T$. Straightforward evaluation of $p(\omega_t)$ needs to solve the Helmholtz equation from scratch, which is expensive. Thus, we wish to bypass those solves as many as possible. To this end, we build our algorithm upon the method of *Asymptotic Waveform Evaluation*

Algorithm 2 Frequency Sweep Precomputation for Mode i

Require: the damped natural frequency ω_i of mode i

procedure AdaptiveFrequencySweep(ω_i)
  Frequency range $\mathcal{R} \leftarrow [\omega_i - \Delta\omega, \omega_i + \Delta\omega]$
  Uniformly sample $\mathcal{R}$ with ω_t s.t. $\omega_1 > \omega_2 > \ldots > \omega_T$
  $\omega^* = \omega_1$
  Construct AWE coefficients at ω^* (i.e., $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ in (3.15))
  for all ω_t in descending order **do**
    if ω_t not in the convg. radius of ω^* (using (3.17)) **then**
      $\omega^* \leftarrow \omega_t - (\omega^* - \omega_{t+1})$
      Construct AWE coefficients at ω^*
    end if
    Compute $p_j(\omega_t)$ at key positions using (3.15) and (A.5)
  end for
  for all key position j **do**
    Build Prony series representation (3.10)
  end for
end procedure

(AWE) [Pillage and Rohrer, 1990; Gallivan *et al.*, 1994] and perform adaptive Helmholtz solves.

Our key idea is to sweep the frequency range $\mathcal{R}$ with multiple steps. At each step, we choose a reference frequency ω_0 and build a local asymptotic expansion of the frequency-varying Helmholtz solution. At the next step, we choose a new reference frequency that cannot be covered by the estimated convergence radius of the expansion at ω_0 in the previous solve. We repeat the step until the entire $\mathcal{R}$ is covered by the convergence ranges of all expansions (see an outline in Algorithm 2).

Boundary Element Solve We use BEM to solve the Helmholtz equation at every reference frequency. For the exterior Helmholtz radiation problem as in our case, The conventional boundary integral equation (CBIE) has non-unique solutions at certain *fictitious frequencies* [Matsumoto *et al.*, 2010]. This will cause serious problems as we need to sweep through a wide frequency range, which likely covers those fictitious frequency values. Instead, we follow the Burton-Miller method [Burton and Miller, 1971], which solves a linear combination of CBIE and a hypersingular boundary integral equation (HBIE) to overcome the non-uniqueness (see Figure 3.12 for numerical validation). We refer the reader to Section A.2 for our implementation details. Ultimately, we solve a dense linear system

$$\mathsf{A}(\omega)\boldsymbol{\phi}(\omega) = \boldsymbol{b}(\omega). \tag{3.11}$$

Here we explicitly express the system with a frequency parameter ω to emphasize its dependence on the frequency value that we are sweeping in $\mathcal{R}$. The solution $\boldsymbol{\phi}(\omega)$ is a vector stacking the acoustic transfer value on object surface elements. With this solution, the transfer value $p(\boldsymbol{x}_j)$ at a key position $\boldsymbol{x}_j$ is computed using the *Kirchhoff integral formula* detailed in (A.5) of Section A.2.

Asymptotic Waveform Evaluation After a BE solve at a frequency ω_0, we have $\bar{p}(\omega_0)$ that satisfies the linear system $\mathsf{A}(\omega_0)\bar{p}(\omega_0) = \boldsymbol{b}(\omega_0)$. Then, a polynomial asymptotic expansion of $\bar{p}(\omega)$ can be built in a local region centered at ω_0,

$$\bar{p}(\omega) = \sum_{i=0}^{N} \bar{p}_i(\omega - \omega_0)^i, \tag{3.12}$$

where $\bar{p}_0 = \bar{p}(\omega_0)$ and $\bar{p}_i, i = 1, \ldots, N$ are coefficients to be determined. To compute $\bar{p}_i$, we take the derivatives of both (3.11) and (3.12) with respect to ω and form a linear system of $\bar{p}_1$,

$$\mathsf{A}(\omega_0)\bar{p}_1 = \boldsymbol{b}'(\omega_0) - \mathsf{A}'(\omega_0)\bar{p}(\omega_0). \tag{3.13}$$

Here $\mathsf{A}'(\omega_0)$ and $\boldsymbol{b}'(\omega_0)$ can be computed analytically (see Section A.4 for details), and we have factorized $\mathsf{A}(\omega_0)$ when solving $\mathsf{A}(\omega_0)\bar{p}(\omega_0) = \boldsymbol{b}(\omega_0)$. Therefore only a fast back-substitution is needed here. Higher-order coefficients $\bar{p}_i$ can be solved in a similar manner using higher-order derivatives of (3.11) and (3.12). We defer the detailed derivation in Section A.3. In short, we have the following equation to solve for $\bar{p}_i$,

$$n!\mathsf{A}(\omega_0)\bar{p}_n + \sum_{i=1}^{n}(n-i)!C_n^i\mathsf{A}^{(i)}(\omega_0)\bar{p}_{n-i} = \boldsymbol{b}^{(n)}(\omega_0). \tag{3.14}$$

where $C_n^i = \frac{n!}{i!(n-i)!}$ are the binomial coefficients. This is a linear system of the form $\mathsf{A}(\omega_0)\bar{p}_n = \boldsymbol{c}$, since all the $\bar{p}_i, i = 0, \ldots, n-1$ are known from previous computation. And again we quickly solve this system by reusing the factorization of $\mathsf{A}(\omega_0)$. After all $\bar{p}_i, i = 0, \ldots, N$ are solved, we can quickly compute the Helmholtz solution at a frequency ω using (3.12).

Extending Convergence Radius with Padé Approximant One drawback of the straightforward polynomial expansion (3.12) is that it tends to have a limited convergence radius (see Figure 3.6). Consequently, we need many AWE solves to cover the frequency range $\mathcal{R}$. To alleviate this problem, we propose to build a Padé approximant, which is known to provide a larger convergence radius than a polynomial expansion although it is derived from polynomial coefficients [Karlsson, 1976]. In particular, provided a set of solved polynomial expansion coefficients $\bar{p}_i, i = 0, \ldots, N$, we match the polynomial expansion (3.12) with a rational polynomial,

$$\sum_{i=0}^{L+M+1}\bar{p}_i(\omega-\omega_0)^i = \frac{P_L(\omega-\omega_0)}{Q_M(\omega-\omega_0)} = \frac{\sum_{i=0}^{L}\boldsymbol{\alpha}_i(\omega-\omega_0)^i}{1+\sum_{j=1}^{M}\boldsymbol{\beta}_j(\omega-\omega_0)^j} \tag{3.15}$$

where both $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are vectors with the same length as $\bar{p}_i$; the quotient is computed using a component-wise division. In practice, we set the rational polynomial orders, $M = \lfloor N/2 \rfloor$ and $L = N - M$,

so this Padé approximant has the same complexity as the polynomial expansion (3.12). We solve $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ by multiplying both sides of (3.15) by $Q_M(\omega - \omega_0)$ and matching the coefficients for all orders of terms. This amounts to solving

$$\begin{bmatrix} \bar{p}_L & \bar{p}_{L-1} & \cdots & \bar{p}_{L-M+1} \\ \bar{p}_{L+1} & \bar{p}_L & \cdots & \bar{p}_{L-M+2} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{p}_{L+M-1} & \bar{p}_{L+M-2} & \cdots & \bar{p}_L \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \boldsymbol{\beta}_2 \\ \vdots \\ \boldsymbol{\beta}_M \end{bmatrix} = - \begin{bmatrix} \bar{p}_{L+1} \\ \bar{p}_{L+2} \\ \vdots \\ \bar{p}_{L+M} \end{bmatrix}. \tag{3.16}$$

Let D denote the number of boundary elements and also the length of $\bar{p}_i$. The above equation describes D independent linear systems, each corresponding to a single component of $\bar{p}_i$ and $\boldsymbol{\beta}_i$; therefore we can solve all D linear systems in parallel. Once $\boldsymbol{\beta}_i$ is obtained, we compute $\boldsymbol{\alpha}_i$ using $\boldsymbol{\alpha}_i = \sum_{j=0}^{i} \boldsymbol{\beta}_j \bar{p}_{i-j}$. Again the product $\boldsymbol{\beta}_j \bar{p}_{i-j}$ indicates a component-wise multiplication. Figure 3.6 illustrates the improvement of the convergence radius.

Adaptive Helmholtz Solves With a depiction of our AWE solver in place, we now present our algorithm to sweep through $\mathcal{R}$ and adaptively perform AWE solves. Let $\omega_{t\ t=1,\ldots,T}$ denote our uniform frequency samples sorted in descending order. Our goal is to evaluate $p(\omega_t)$ at all key positions. We start from the highest frequency sample ω_T and build an AWE expansion series at ω_T. Then we move on to ω_{T-1} and check if ω_{T-1} is within the convergence radius of the series at ω_T. If it is, we directly evaluate the series to compute $p(\omega_{T-1})$ and continue to the next sample ω_{T-2}. At some point, a sample ω_i is out of the convergence radius, and thus the obtained AWE series becomes invalid. At this point, we know a lower bound of the convergence radius of the series at ω_T is $\omega_T - \omega_{i+1}$. We also observe that the convergence radius of the AWE series increases as the expansion frequency decreases. Therefore, we build a new AWE series at the frequency sample $\omega_j = \omega_i - (\omega_T - \omega_{i+1})$. Since ω_j is smaller than ω_T, we guarantee that ω_i is now within the convergence radius of the series

at ω_j. From there we switch to the new series at ω_j and continue our transfer evaluation. We repeat these steps until the entire frequency samples $\omega_t, t = 1, \ldots, T$ are evaluated (see Algorithm 2). The main advantage of this process is that we solve the AWE only at a few automatically selected frequency samples while relying on the expansion form to quickly evaluate transfer values for all the samples.

A simple approach to check if a frequency ω is within the convergence radius is to evaluate the series at ω and substitute it back into (3.11) to compute the residual. However, this approach needs to construct the dense matrix $\mathsf{A}(\omega)$ for every check. Instead, we propose a faster algorithm by exploiting a mathematical insight of Padé approximant: two consecutive orders of Padé solutions are very close inside the convergence radius, but they diverge rapidly when this radius is reached. To harness this insight, we compute

$$\hat{\bar{p}}(\omega) = \frac{P_{L-1}(\omega - \omega_0)}{Q_{M-1}(\omega - \omega_0)} \text{ and } \bar{p}(\omega) = \frac{P_L(\omega - \omega_0)}{Q_M(\omega - \omega_0)}, \tag{3.17}$$

and require $\|\hat{\bar{p}}(\omega) - \bar{p}(\omega)\| \leq \epsilon$. For all our examples, we use $L = 6$, $M = 5$ and $\epsilon = 10^{-4}$. On average we only need about 5 AWE solves to cover a frequency range of 5kHz to achieve this error tolerance (See Figure 3.6).

3.4.3 Frequency-Adaptive Mesh Simplification

To further speed up the precomputation, we accelerate each Helmholtz solves by adaptively simplifying object surface meshes. It is well known that the complexity of BEMs depend on the number of surface elements N. The smaller N is, the faster computation can be. For Helmholtz solves, it is also found that the element size should be bounded by the wavelength [Jerri, 2005]. Corresponding to human hearing range from 20Hz to 20kHz, the wavelength varies from 17 meters to 1.72 centimeters. Thus we can use fewer elements while retaining the accuracy for lower frequencies.

Furthermore, we observe that the scale of spatial variance of an object's mode shapes is often much larger than the size of the mesh elements (see Figure 3.7). As a result, the modal displacement data can be well resolved even with a coarse surface mesh.

We therefore use frequency-adaptive surface discretization in our BE solves. We start from a fine surface mesh sufficient for the highest frequency (e.g., 20kHz). For each mode i, surface vertices are also associated with their modal displacement vectors $\tilde{\boldsymbol{u}}_i$ extracted from the shape matrix U. Next, we divide our interested frequency range $\mathcal{R} = [\omega_0 - \Delta\omega, \omega_0 + \Delta\omega]$ into multiple intervals, each with a fixed frequency band. In practice, we use a 2kHz frequency band and hence up to 3 intervals for $\mathcal{R}$. For each frequency interval and each mode i, we construct a simplified surface mesh as well as corresponding modal displacement data. We perform the mesh simplification at the beginning of our precomputation stage. During our Helmholtz solves performed in §3.4.2, we adaptively choose the mesh resolution and modal displacement data based on the target frequency range and mode.

Edge Collapse Algorithm We build our mesh simplification algorithm based on the edge collapse algorithm of Hoppe [1999], and follow their notations therein. Each mesh vertex v has a 6D vector $\boldsymbol{v} = [\,\boldsymbol{p}^T\ \boldsymbol{u}^T\,]^T$, where $\boldsymbol{p}$ is the vertex position, and the modal displacement vector $\boldsymbol{u}$ is used as a vertex attribute. The quadric error function for collapsing an edge is defined as

$$Q^v(\boldsymbol{v}) = \sum_{f \in \mathcal{N}(v)} A(f) \left(Q_p^f(\boldsymbol{v}) + Q_u^f(\boldsymbol{v}) \right)$$

where $A(f)$ is the area of a triangle f adjacent to $\boldsymbol{v}$, $Q_p^f(\boldsymbol{v})$ measures the distance of $\boldsymbol{p}$ to the plane containing f, and $Q_u^f(\boldsymbol{v})$ measures the deviation of $\boldsymbol{u}$ from a linearly changing modal displacement field on the triangle f. Both terms are simply zero-extended versions of those in [Hoppe, 1999]. We therefore refer the reader to that paper for details. In short, $Q(\boldsymbol{v})$ has a 6D quadratic form,

$Q^v(\boldsymbol{v}) = \boldsymbol{v}^T \mathsf{A}\boldsymbol{v} + 2\boldsymbol{b}^T\boldsymbol{v} + \boldsymbol{c}$. When collapsing an edge connecting v_1 and v_2 into a new vertex v_{new}, we solve for its position $\boldsymbol{p}_{new}$ and modal vibration vector $\boldsymbol{u}_{new}$ by minimizing

$$\begin{aligned} \boldsymbol{v}_{new} &= \arg\min_{\boldsymbol{v}} Q^{v_1}(\boldsymbol{v}) + Q^{v_2}(\boldsymbol{v}) \\ \text{s.t.} \quad & \boldsymbol{g}_{vol}^T \boldsymbol{p} + d_{vol} = 0. \end{aligned} \tag{3.18}$$

Here $\boldsymbol{g}_{vol}$ and d_{vol} are respectively a 3D vector and scalar determined by the 1-ring local geometry of the collapsing edge. This linear constraint is to ensure volume preservation.

Volume Velocity Preservation While previous methods can achieve volume preservation, they shrink the modal amplitude in the process of edge collapse, resulting in a loss of sound power (see Figure 3.8). We address this problem by introducing a constraint on the object's *volume-velocity*. Given a modal displacement vector $\boldsymbol{u}$ at a frequency ω, the object's volume velocity is defined as

$$\int_S (\boldsymbol{u} \cdot \boldsymbol{n}) \omega e^{i\omega t} dS = \omega e^{i\omega t} \int_S \boldsymbol{u} \cdot \boldsymbol{n} dS.$$

This quantity has been previously used as a far-field approximation of sound power [Johnson and Elliott, 1995]. Our goal is to ensure its preservation during our mesh simplification. When an edge collapses into a vertex v, only the volume-velocity contributed by its 1-ring triangle fan can be changed. Ignoring the unchanged time-harmonic part and assuming a piece-wise constant modal vibration, we have the volume-velocity constraint,

$$\frac{1}{6} \sum_{f \in \mathcal{N}(v)} \left[(\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2}) \right]^T (\boldsymbol{u} + \boldsymbol{u}_{f1} + \boldsymbol{u}_{f2}) = C_v \tag{3.19}$$

where f denote a triangle adjacent to v, $(\boldsymbol{p}, \boldsymbol{p}_{f1}, \boldsymbol{p}_{f2})$ are f's vertex positions, $(\boldsymbol{u}, \boldsymbol{u}_{f1}, \boldsymbol{u}_{f2})$ are the modal displacement vectors on those vertices, and C_v is a constant volume velocity value computed from the corresponding triangle region before the edge collapse. Now we need to minimize

$\bar{Q}^v(\boldsymbol{v}) = Q^{v_1}(\boldsymbol{v}) + Q^{v_2}(\boldsymbol{v})$ with two constraints when collapsing an edge,

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}} \bar{Q}^v(\boldsymbol{v}) \quad \text{subject to constraint (3.18) and (3.19).}$$

This is a quadratically constrained quadratic programming (QCQP) problem, generally considered to be NP-hard. Fortunately, in this particular QCQP problem, because the quadratic constraints involve only $\boldsymbol{p}^T\boldsymbol{u}$ but not $\boldsymbol{p}^T\boldsymbol{p}$ or $\boldsymbol{u}^T\boldsymbol{u}$, we are able to solve it using iterations of linearly constrained problems: we start from an initial guess of $\boldsymbol{v}$ by minimizing $Q^v(\boldsymbol{v})$ without any constraints, and this amounts to solving a 6D linear system, $\mathsf{A}\boldsymbol{v} = \boldsymbol{b}$. Then we iteratively apply a staggered sequence of two quadratic optimization solves

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}} Q^v([\boldsymbol{p}^T\ \boldsymbol{u}^T]^T) \ \text{ subject to (3.19) only,} \tag{3.20}$$

$$\boldsymbol{p} = \arg\min_{\boldsymbol{p}} Q^v([\boldsymbol{p}^T\ \boldsymbol{u}^T]^T) \ \text{ subject to (3.18) and (3.19).} \tag{3.21}$$

In the first solve (3.20), we use vertex positions $\boldsymbol{p}$ from previous iterations and compute $\boldsymbol{u}$. In the second solve (3.21), we fix displacement vector $\boldsymbol{u}$ using values resulting from (3.20) and compute vertex positions. Both solves minimize a quadratic form with linear equality constraints. We solve them using the method of Lagrange Multipliers: problem (3.20) becomes a 4D linear system, while problem (3.21) amounts to a 5D linear solve (see Section A.5). In practice, only tens of iterations are needed for convergence. As demonstrated in Figure 3.9, using the adaptively simplified meshes greatly speeds up the boundary element solves (in §3.4.2), while introducing very little numerical error.

Example	(i) Complexity		(ii) Mesh Simplification						(iii) Adaptive Freq. Sweep			(iv) Runtime Evaluation				
	# tet.	# modes	before (avg.) # tri.	before (avg.) BE Solve	after (avg.) # tri.	after (avg.) BE Solve	simp. time	speedup	before # solves	after # solves	speedup	before size	before time	after size	after time	speedup
Plate	404k	59	100k	19m	5750	4.2m	16.8m	4.2×	4740	253	17.2×	8.1MB	59m	5.1MB	12.9s	274×
Mug	292k	61	68k	35m	7255	6.1m	14.7m	5.5×	4492	379	11.3×	8.7MB	96m	5.4MB	13.6s	424×
Bunny	130k	56	30k	52m	4297	4.6m	10.2m	10.1×	3360	198	14.5×	7.7MB	132m	4.8MB	22.2s	356×
Bottle (solid)	160k	197	35k	21m	4139	4.1m	30.6m	4.9×	13396	1068	10.4×	30.2MB	237m	22.1MB	28.9s	492×
iJump	62k	70	14k	14m	3123	3.8m	6.5m	3.7×	5075	267	17.6×	9.2MB	96m	6.0MB	24.8s	232×
Stairs	78k	49	12k	14m	5425	5.8m	21.2m	2.9×	3626	221	13.2×	6.7MB	38m	4.2MB	11.6s	197×
Oloid (shell)	-	300	32k	29m	7841	5.6m	28.4m	4.9×	12623	715	17.1×	62.4MB	258m	26.1MB	12.2s	1270×
Cow (shell)	-	300	65k	42m	6406	5.1m	40.3m	7.8×	14131	624	22.2×	61.2MB	312m	25.7MB	23.8s	785×
Bottle (shell)	-	200	35k	23m	5364	4.7m	36.6m	4.4×	9246	436	20.5×	42.7MB	186m	19.9MB	19.4s	575×

Table 3.1: **Statistics and Timings**: (i) the size of tetrahedral meshes and modes; (ii) the averaged number of triangles before and after mesh simplification, the averaged BE solve time with and without simplification, the mesh simplification time, and the speedup to compute transfers of all modes, (iii) the total number of Helmholtz solves without and with adaptive frequency sweep, the speedup achieved using adaptive frequency sweep with simplified meshes. (iv) the memory overhead for transfer evaluation without and with key-position least-squares solves, the timings of transfer update using standard BE solves on a 20-core cluster, the timings of transfer evaluation using our approach on a quad-core desktop, and the computational speedup. Note: the memory without key-position least-squares solves only represents the storage on a single frequency. This storage increases as we sweep through the frequency range $\mathcal{R}$, whereas our model uses a fixed memory.

3.5 Validation of Interactive Acoustic Transfer

Performance We profiled the performance of our algorithm, and summarized speedups of each step over the straightforward approaches, as well as the runtime speedups. Table 3.1 lists the statistics of our examples. The precomputation timings were measured on a 20-core Intel Xeon E5 cluster, and the runtime profiling was performed on a desktop with a quad-core Intel Xeon E5 (3.4GHz) CPU. Due diligence has been taken to exploit multi-core parallelization for both the precomputation and runtime sound synthesis. On average, our adaptive mesh simplification achieved $5\times$ speedups for Helmholtz solves; our adaptive frequency sweep led to at least $10\times$ speedups; and at runtime, given user-specified parameters, we are able to synthesize sound with more than $300\times$ speedups over the traditional approach which needs to recompute the Helmholtz solutions. We note that it

is possible to further boost runtime performance for interactive parameter editing: for example, at runtime one can start a background thread performing the least-squares solves of multipole coefficients at more densely sampled frequencies while the user is adjusting parameters, and cache the computed transfer coefficients for later reuse.

The additional memory size needed for runtime use of our Prony representation for all key positions is less than 25MB per model. We also note that the major memory bottleneck of a modal sound model is the storage of shape modal matrix U that can take hundreds of megabytes of memory, depending on the mesh resolution. Our frequency-sweeping transfer representation is resolution-independent, and adds little memory overhead. We note a recent method [Langlois *et al.*, 2014] that compresses the modal matrix U and complements to our approach.

Comparisons To demonstrate the effects of transfer values on final sounds and the accuracy of our transfer evaluation, we compared the sounds computed without transfer, with transfer at a fixed frequency, with transfer using our model, and the exact transfer using BE solves. For this comparison, we also chose different modal sound parameters to generate different sound effects. We observed that the resulting sounds from our model are very close to the sounds using brute-force transfer evaluation, while the sounds without transfer and with constant transfer both show audible differences from the ground-truth sounds. Please see the accompanying video for animations and sound comparisons.

Numerical Validation We further validated our models numerically. Our runtime transfer evaluation are approximated by least-squares problems formulated using key-position transfers. In Figure 3.10, we validate its accuracy by comparing with the results from full BE solves. For low-frequency Helmholtz solves (Figure 3.10 left), our results agree with the brute-force solution very well. As expected, the high-frequency solves (Figure 3.10 right) are numerically more challenging,

and our approximation degrades. However, Figure 3.11 shows the convergence of our approximation as the number of key positions increases. Therefore, we can always increase the accuracy of our runtime approximation by adding more key points. This feature provides the user easy control of the performance-accuracy tradeoffs for specific applications.

Lastly, Figure 3.12 validates our BEM implementation with the conventional CBIE approach. As shown, our implementation based on the Burton-Miller method [Burton and Miller, 1971] is more robust and agrees with the analytical solution very closely.

3.6 Results

3.6.1 Sound Editing Examples

Our interactive transfer estimation enables flexible and efficient approaches to tweak modal sound parameters, explore different sound characteristics, and achieve desirable sound effects. We now demonstrate with three applications. All the animations are simulated using [Kaufman *et al.*, 2008] except iJump is from [Tan *et al.*, 2012]. Please see the accompanying video for full results.

Fast Parameter Editing Modal sound models are often used to synthesize sounds automatically synchronized with simulated animations. To achieve certain sound characteristics, the user might start with physical parameters of target materials. However, even for a single material, its material parameters are given in a range. For instance, polyethylene, a common plastic, has a Young's modulus in a range from 0.11GPa to 0.45GPa, which doubles the modal frequencies when changed from the lower end to the upper end. In addition, there are no mechanically well-defined damping parameters [Adhikari and Woodhouse, 2001], although the damping can significantly affect the sound perception [Klatzky *et al.*, 2000]. Consequently, one has to rely on a trial-and-error approach

to tune the parameters. It is therefore desirable to have a fast sound synthesis method to shorten the tuning cycle. In our examples, we take a rigid-body simulation as input, and edit Young's modulus and damping ratio to synthesize sounds produced by different materials ranging from wood, plastic, porcelain to metal (see Figure 3.13.**h**). Our runtime synthesis time is always less than 30 seconds.

Parameter Space Exploration Our method allows the user to continuously explore the parameter space. In our implementation, we present the user with a 2D parameter space whose two axes are damping scale and stiffness scale respectively (see the video). When the user clicks a point in the coordinate system, we immediately synthesize the sounds with corresponding stiffness and damping values and present to the user Figure 3.1. Take Stairs as a demonstration. With a single pass of precomputation, we explore the parameter space, and identify a set of parameters that produces different pitches corresponding to a set of music notes with different timbres. After we are satisfied with the resulting sound characteristics, we use the parameters to generate sounds of more complex animations. In Stairs, we choose three different materials and produce sounds that match the melody "Song of the Wind" (see Figure 3.13.**h**).

Thin-Shell Modal Models Our method is not limited to editing solid modal sound models. We also apply our method to edit thin-shell modal sound models. In the precomputation, we compute thin-shell modal matrices and vibration frequencies following the method proposed by Chadwick et al. [2009b]. The rest of the pipeline is exactly the same as the solid modal sound model. In the accompanying video, we demonstrate different thin-shell sound effects edited using our method (see Figure 3.13.**e**, **f** and **g**).

Extension: Time-varying Frequency Effects Finally, we extend our method to allow the user to specify time-varying parameters, we can thus approximate sound effects with frequency shift, which

is usually caused by nonlinear modal vibrations. This extension is straightforward: with user-guided time-varying parameters, we sample the values across the temporal domain and evaluate transfers for smooth interpolation. The modal vibration equation (3.3) with time-varying coefficients is still integrated using Runge-Kutta method as presented in §3.3.1. In our examples, we used the time-varying stiffness scale to mimic the nonlinear pitch changes, such as pitch gliding [Penttinen *et al.*, 2006] (see Figure 3.13.**c**). We also explored an example in which the user specifies nonphysical time-varying frequencies to produce interesting artistic effects such as the one in iJump (see Figure 3.13.**d**).

3.6.2 Preliminary User Studies

Experiment Setup We perform four user studies to evaluate the perceptual quality of different levels of transfer approximation accuracy in our method.

1. We generate sounds using four different sets of hit locations and impact forces. For each setting, we compute three versions of sounds, denoted as A, B, and C, using different approximation accuracies. The accuracy is measured as the normalized least-squares residual as used in §3.5 (Table 3.2). We then perform the Two-alternative forced choice (2AFC) tests: we generate three pairs of sounds, AB, BC and AC, for every set of sounds, and present the human subjects with each pair of sounds, one immediately after the other, along with a reference sound generated without transfer approximation. The subjects are asked which sound in each pair is closer to the reference.

2. We ask the subjects to rank the similarity of pairs of sounds, one approximated sound from above and one reference audio, on a Likert scale (i.e., choosing from "very similar", "similar", "neutral", "different", and "very different"). We repeat this experiment for all sound settings computed in previous study. This study is to examine preliminarily the correlation between

the numerical errors and the perceived difference.

3. Then, for each sound generation setting, we choose different number of samples to estimate transfer values. This results in sounds with different accuracies (measured by the normalized L_2 error in Table 3.3). We present the subjects with each of these sounds and a fully simulated sound, along with the reference sound. We then ask them which sound (the approximated sound or the fully simulated sound) is closer to the reference.

4. Lastly, we validate our assumption of using fixed modal shapes. For each test example, we computed two sounds, one with varying modal shapes and one with fixed modal shapes. We use the BUNNY example with three materials (wood, porcelain and metal). When building the modal sound model with fixed modal shapes, we use our method to simplify meshes. While the mesh simplification is frequency dependent, on average we observed more than $20\times$ reduction of the number of triangles. We then ask the subjects to rate the similarity of the two sounds on a Likert scale. We present these sounds in a random order to avoid possible bias from ordering.

Analysis of the Results We conducted the experiments with 40 subjects, ranging from age 20 to 31, with 35% female and 65% male. All subjects are university affiliates who reported no problem with hearing and had no sound design experience before. As part of the training process, we show each subject a set of sample questions and walk through the interface.

In the first pairwise comparisons, we aggregated the results from all examples. Overall, 86.6% of the subjects thought the full-sample sound was more similar to the reference sound among all three sounds. Using half and a quarter of the samples won 47.8% and 15.6% of user selections, respectively. (see Table 3.2).

Figure 3.14 visualizes the results of the second experiment. For each sound (of A, B and C), we plot

sound	A	B	C
# samples	100%	50%	25%
averaged error	10^{-9}	0.299	0.641
winning percentage	86.6%	47.8%	15.6%

Table 3.2: **Statistics of the first user study.** The error is measured as the normalized least-squares residual. As the error increases, fewer and fewer subjects consider the approximated sounds to be similar to the reference.

# samples	100%	80%	60%	40%	25%
error	10^{-9}	0.08	0.19	0.37	0.53
winning percentage	47.5%	32.5%	5%	2.5%	2.5%

Table 3.3: **Statistics of the third user study.** As the error increases, more and more subjects can perceive the difference between the reference sound and the approximated sound.

its frequencies of being classified on each Likert scale category. We found that as the approximation error increases, it becomes easier for the subjects to notice the difference between the approximated sounds and the reference.

The third study shows that when the error is very small, the subjects cannot tell the difference. As the L_2 error is slightly increased to 0.08, 32.5% of the subjects perceived the approximated sound to be similar to the reference sound. As the number of samples drops below 60%, we observed a clear decline in the perceived similarity. In other words, subjects were able to discern the difference once the errors were above 0.08 (see Table 3.3).

Lastly, in the fourth user study, 82.5% of the subjects considered two sounds with fixed and varying modal shapes to be "Very similar", and 17.5% of them chose "Similar". This suggests that using fixed modal shapes for fast transfer approximation is indeed plausible.

3.7 Conclusion

We have presented the method for fast runtime modal sound synthesis via efficient and more general precomputation. It greatly eases parameter tuning for desirable sound effects, and has the ability to generate various sound effects even using a single model. Our efficient runtime synthesis is realized by solving small-scale least-squares problems to estimate multipole coefficients of transfer functions. The least-squares formulation relies on precomputation. To improve its efficiency, we utilize Padé approximant to sample key frequencies adaptively and propose a volume-velocity-preserving mesh simplification algorithm to speed up individual Helmholtz solve. With numerical comparisons and user studies, we demonstrate its use in sound synthesis applications such as fast parameter tuning for various sound effects, and extend it to support the creation of time-varying sound effects. We augment and leverage several numerical techniques throughout, such as Prony's method and Padé approximant, hoping that these tools can be useful in other graphics research areas as well.

Although we have shown that our approximation is comparable to the ground-truth results both numerically and qualitatively, it remains unexplored how far we can go to further speed up the computation. For example, can we take even coarser samples and solve Helmholtz on even simpler meshes while maintaining the perceptual plausibility? In addition, it is well-known that the Helmholtz problem at higher frequencies tends to be more ill-conditioned and thus numerically more challenging. This difficulty is also observed in our experiments, as our least-squares solves in §3.5 can not perfectly agree with the accurate solutions for frequencies higher than 12kHz, and the numerical error of transfer solves (shown in Figure 3.9) becomes larger as the frequency increases. For modal vibration sound, the high-frequency modes have large damping coefficients, and therefore this inaccuracy is hardly noticeable. However, when extending this method for editing other sound models such as fluid sounds, we hope to have a more accurate high-frequency approxima-

tion. In the proposed model we are able to generate different sound effects such as wood, porcelain, metal, etc.. With different input models, the results of linear model analysis, mostly the modal frequencies, are very distinct. As a result, it may require different parameters to achieve similar sound effects. One possible extension is to build a geometry-invariant measure such that a set of parameters can produce similar sound effects regardless of the input model geometry. Moreover, as observed in the OLOID (shell) example, different transfer approaches may produce similar sounds that the users cannot distinguish. We would like to better understand the reason that causes this ambiguity, which might in turn suggest a way to exploit this ambiguity. One common feedback from users is that the stiffness and damping parameters are not very intuitive at the beginning; they only started realizing their different effects during the second or even third trial. Therefore, one possible future work is to identify more intuitive sound model parameters for user adjustment. Finally, another interesting direction is to investigate a combination of our method and traditional Foley sound tools based on sound recording and granular synthesis to circumvent the numerical difficulties at high frequencies.

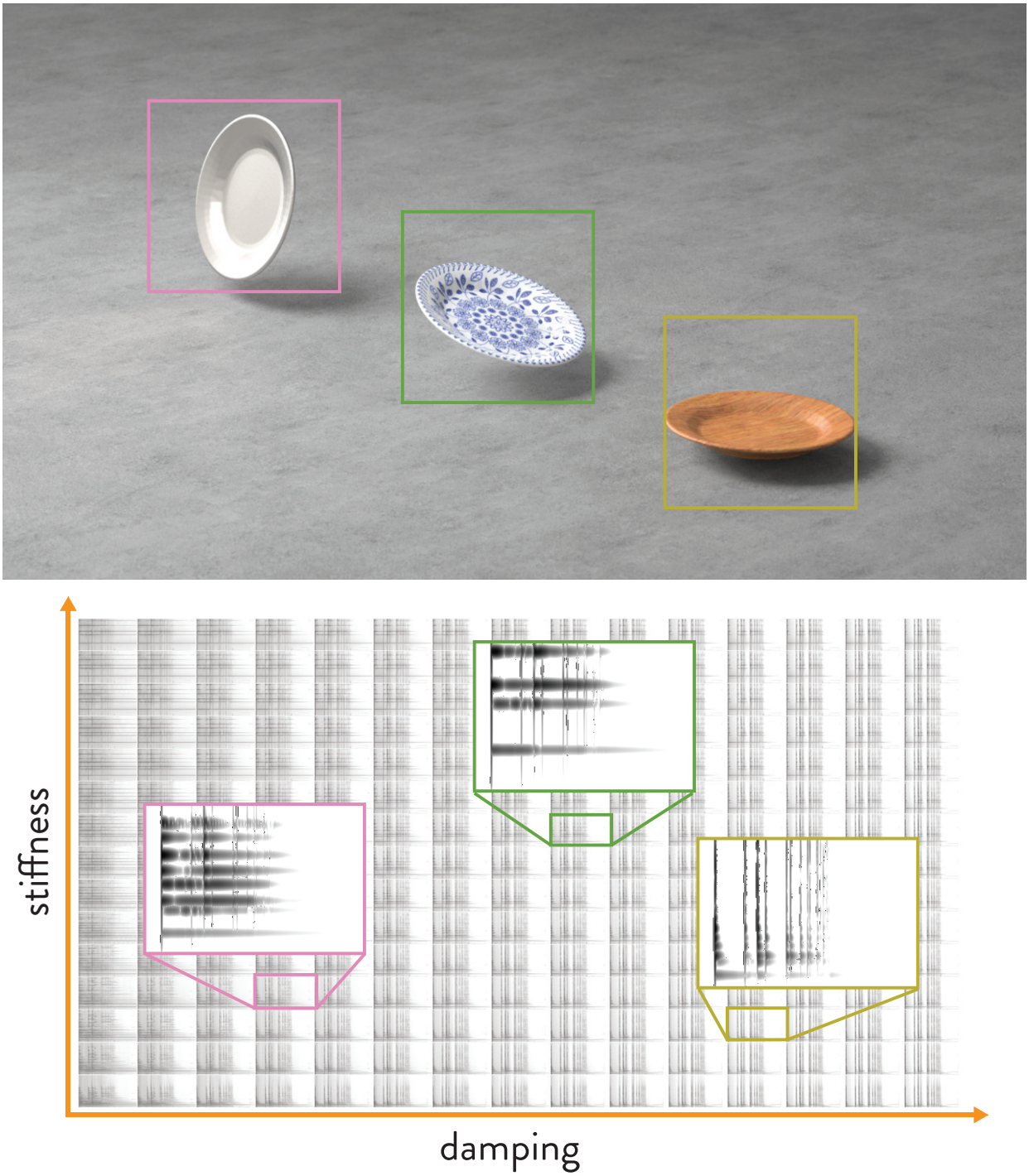


Figure 3.1: **Parameter Exploration using our method:** With our precomputed information, we are able to explore the space of modal sound parameters at runtime, achieving numerous sound effects (bottom) synchronized with a physics-based animation. The three spectrograms highlighted in the colored boxes correspond to (left to right) metal, porcelain, and wood materials shown on the top.

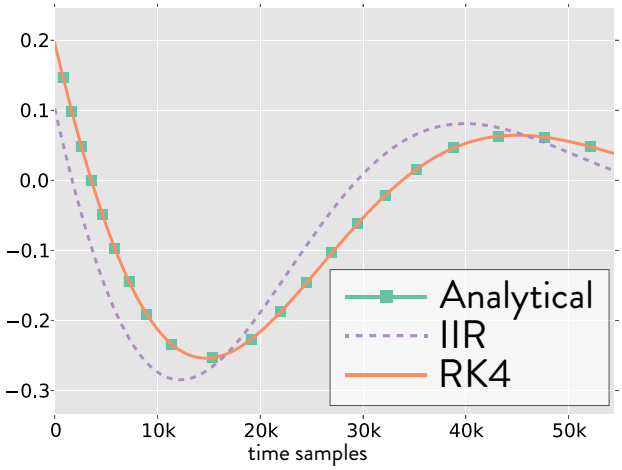


Figure 3.2: **Comparison between Runge-Kutta and IIR filter:** Given a time-varying vibration equation, fourth-order Runge-Kutta (RK4) integrator (orange) offers higher accuracy against the IIR filter (purple), which was used in previous methods.

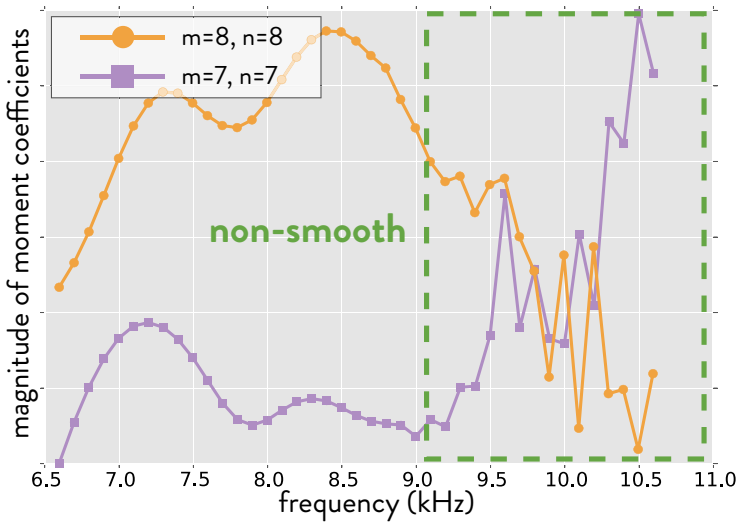


Figure 3.3: **Non-smoothness of M_n^m:** The high-order M_n^m becomes non-smooth and fluctuates strongly at high frequencies, making direct interpolation difficult. We note that these orders (i.e., N=7,8) are necessary in the expansion for sufficient accuracy.

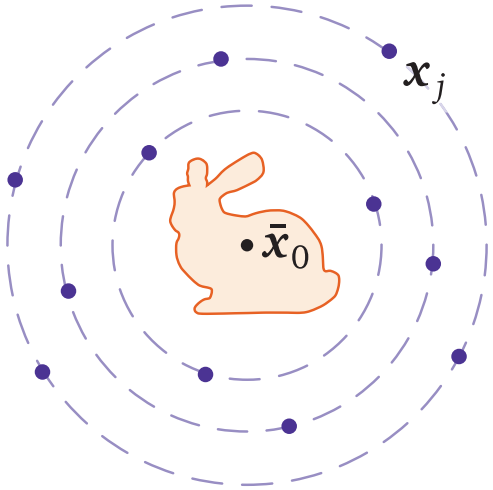


Figure 3.4: Key Positions

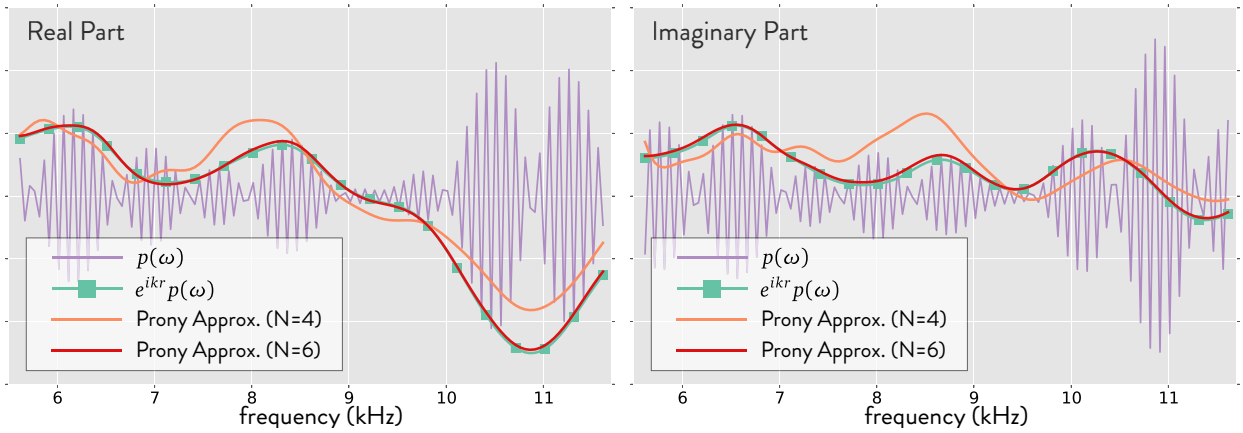


Figure 3.5: **Frequency-Sweeping Transfers:** We choose one mode of the BUNNY model, evaluate $p(\omega)$ using BEM at a fixed point as frequency sweeps and plot both real and imaginary parts. $p(\omega)$ oscillates dramatically (purple); factoring out e^{-ikr} produces a much smoother curve (green); 4th-order Prony series gives a coarse interpolation curve (orange), while 6th-order series produces a curve (red) almost identical to the original function.

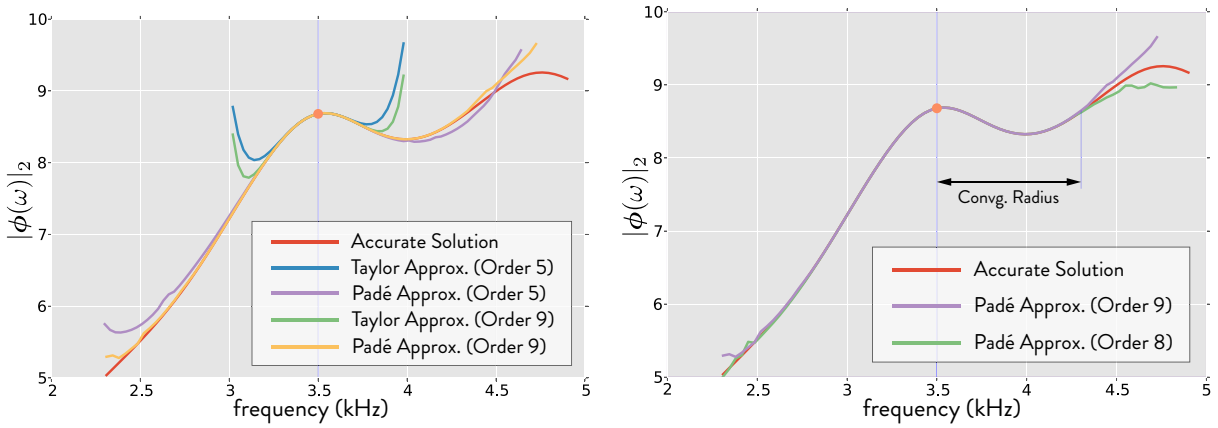


Figure 3.6: **Asymptotic Waveform Evaluation:** Using the BUNNY model, we sweep a frequency range and evaluate $|\phi(\omega)|_2$ in (3.11) using different expansions and accurate Helmholtz BE solves. (left) We compare the convergence radius of polynomial expansions against Padé approximant. Accurate BE solution is plotted in red. Both 5th-order and 9th-order polynomial expansion diverge from the accurate solution faster than their Padé counterparts. (right) 8th-order Padé approximant agrees with the 9th-order one closely until the convergence radius is reached.

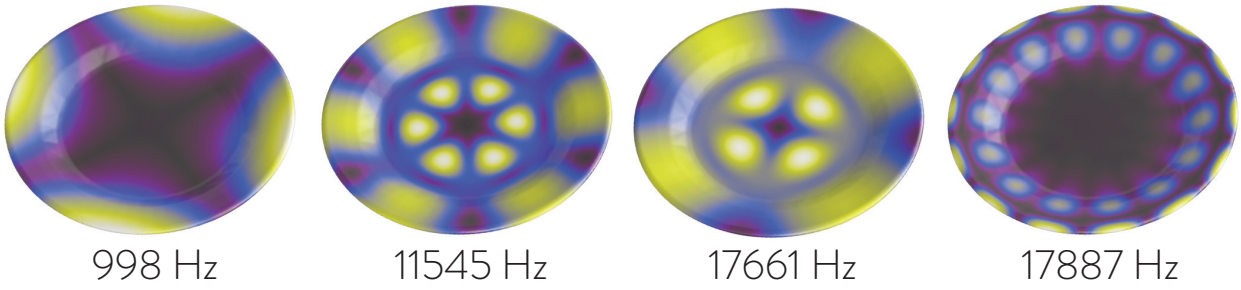


Figure 3.7: **Smooth Modal Shapes:** Color encodes the modal displacement amplitude of the PLATE model; modal frequencies are listed below each subfigure. Even for high frequency modes, their modal displacement varies smoothly on the surface, making it possible to perform mesh simplification.

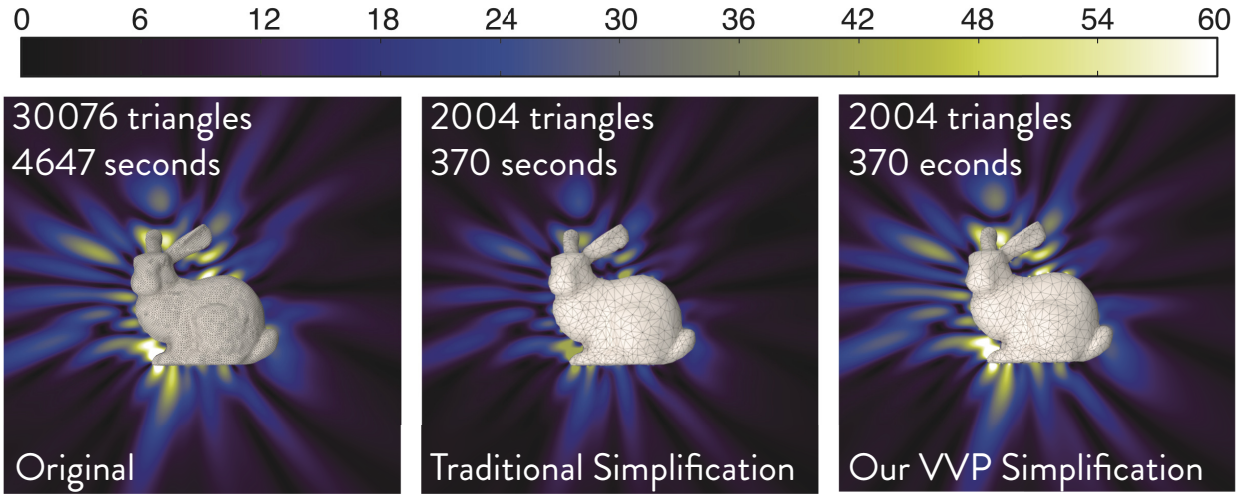


Figure 3.8: **Volume-Velocity-Preserving Mesh Simplification:** We solve the Helmholtz equation using the original high-resolution mesh (left). We then simplify the mesh without volume-velocity preservation (middle) and with volume-velocity preservation (right). For both meshes, the Helmholtz solve is 12.6× faster than the original Helmholtz solve. Without volume-velocity preservation (middle), the acoustic transfer field loses radiation power, while the volume-velocity-preserving mesh simplification (right) results in almost identical pressure field to the original high-resolution solve.

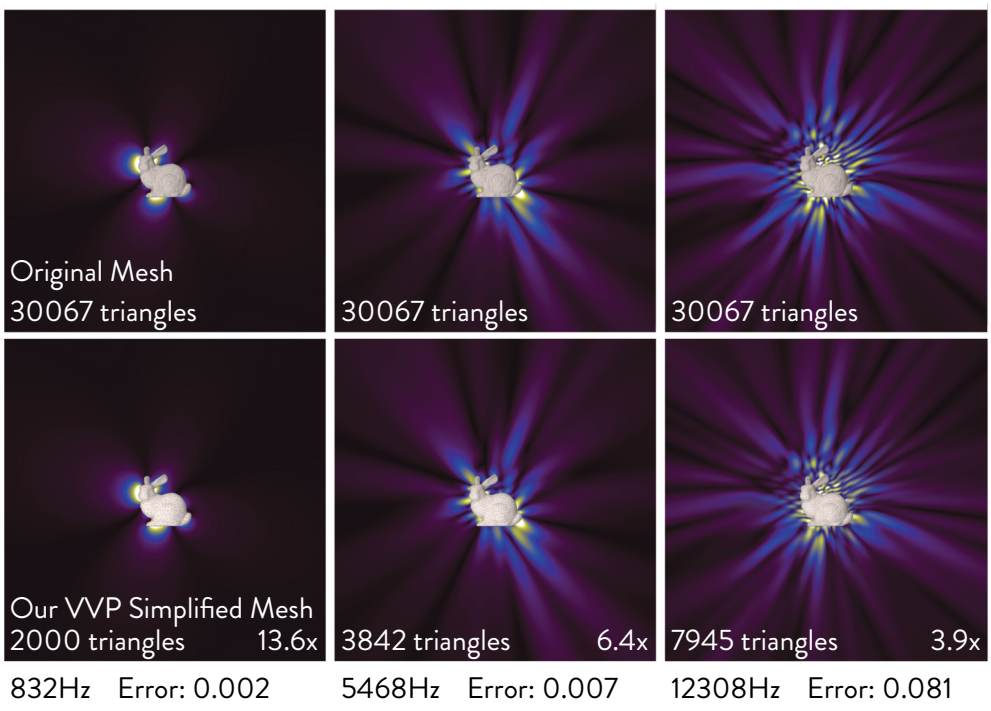


Figure 3.9: **Transfer Values with Frequency-Adaptive Remeshing:** We compare transfer values computed using the original high-resolution mesh (top) with the values using simplified meshes (bottom) for three modes with low, medium and high frequency values. Even for high-frequency modes which require a relatively high-resolution mesh, our method achieves nearly $4\times$ speedup, while retaining a low L_2 error ($< 8.2\%$).

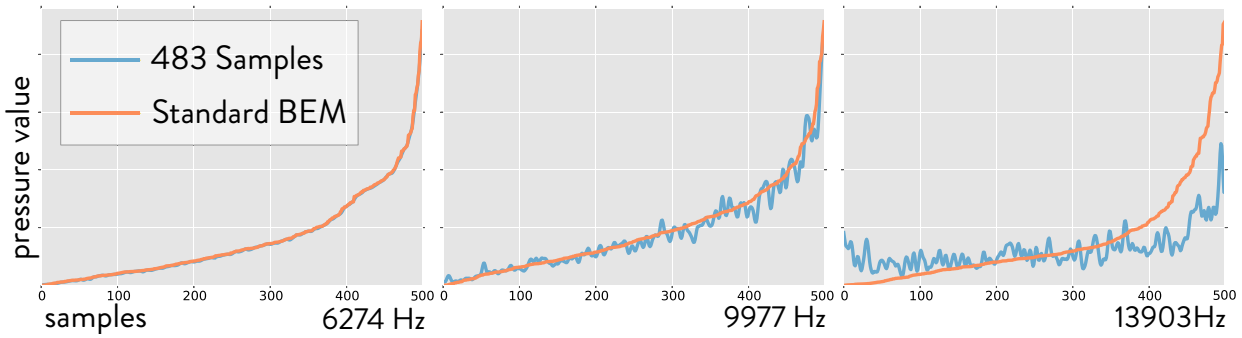


Figure 3.10: **Accuracy of Least-Squares Approximation:** we sample 483 key positions for the least-squares estimation of M_n^m. We estimate M_n^m with three frequency values and use them to evaluate acoustic transfer values at 500 randomly selected locations (blue). Meanwhile, we compute the accurate Helmholtz solution at the same locations (orange). For better visualization, we sort the locations based on their accurate transfer values. As frequency increases, the accuracy of our approximated transfer values degrades gracefully.

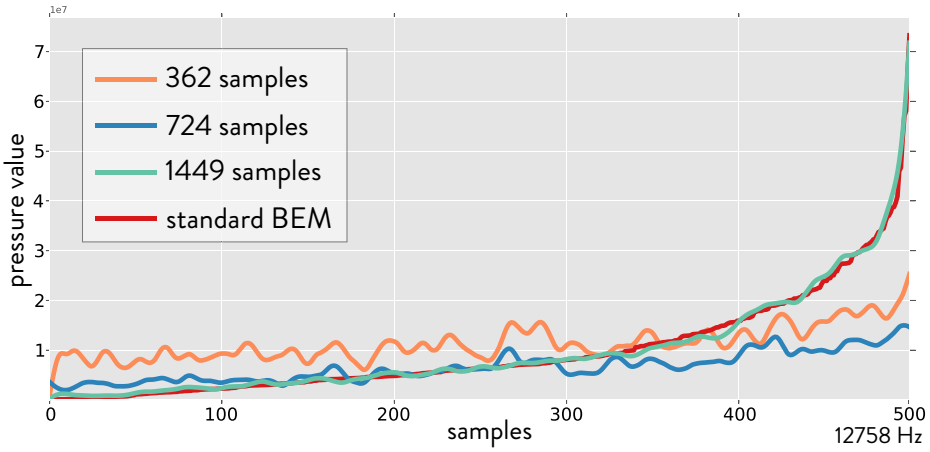


Figure 3.11: **Convergence of Least-Squares Approximation:** Fixing the frequency value $f = 12758Hz$, we estimate M_n^m using an increasing number of key positions. We use the estimated M_n^m to evaluate acoustic transfer values at 500 randomly selected locations (orange, blue, and green curves), and compare them against accurate transfer values solved using conventional BEM (red). As the number of key positions increases, we get higher fidelity for the estimated transfer values.

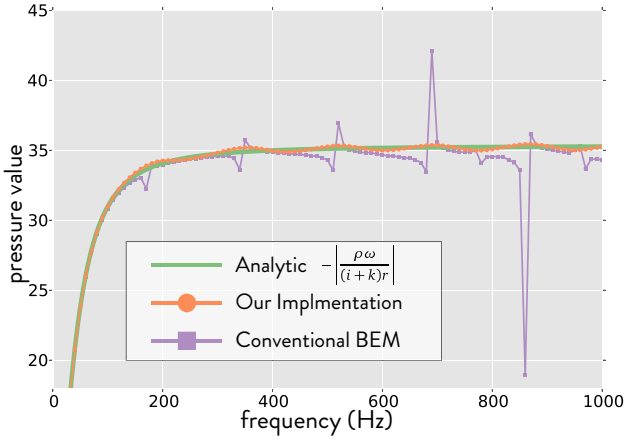
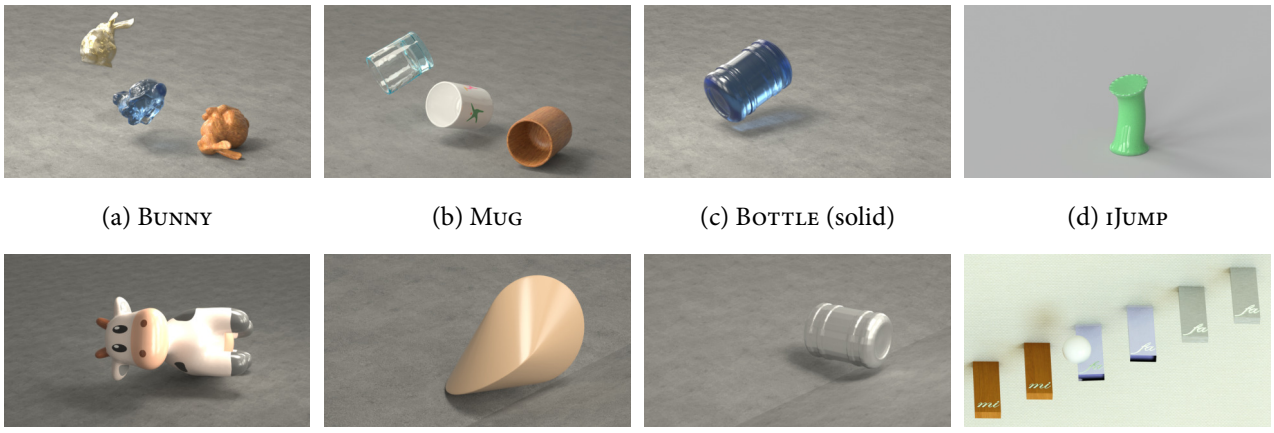


Figure 3.12: **BEM Comparison:** Using a pulsating sphere with known analytic solution, our BE implementation (orange) agrees with the analytic solution (green) as frequency sweeps, whereas the CBIE solver (purple) has large error at fictitious frequencies.



(a) Bunny (b) Mug (c) Bottle (solid) (d) iJump

(e) Cow (shell) (f) Oloid (shell) (g) Bottle (shell) (h) Stairs

Figure 3.13: (**a** and **b**) We edit the Young's modulus and damping values at runtime, and produce sound effects corresponding to various materials. (**c** and **d**) We explore the examples that allow the user to change individual frequency values in a time-varying way, producing nonlinear artistic effects while retaining physical realism. (**e, f** and **g**) We apply our method to edit sound effects of thin shells. (**h**) We explore modal sound parameters so that each stair makes sounds corresponding to a music note, and the entire ball bouncing sequence produces a melody.

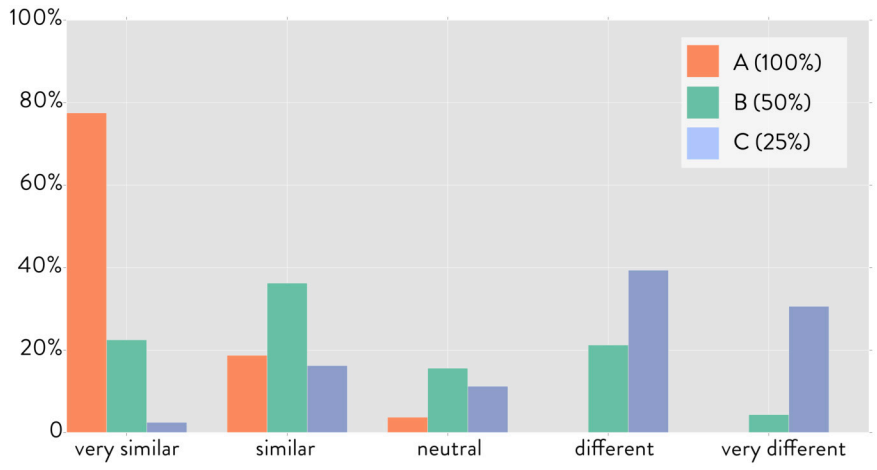


Figure 3.14: **Statistics of the second user study.**

Chapter 4

Acoustic Filters Simulation and Optimization

4.1 Introduction

Acoustic filters have numerous important applications, whether to produce a desired sound pitch or to attenuate undesired noise. These applications, ranging from wind instruments to mufflers and hearing aids, all rely on the same fundamental physical principle: when sound waves pass through a cavity, part of the waves reflect back and forth, effectively boosting or suppressing certain acoustic frequencies. In this process, the filtered frequencies are largely affected by the *shape* of the cavity.

However, for all but the simplest cavity shapes, the influence of the shape on the filtered frequency bands is complicated and unintuitive. Thus, the current process for improving the quality of acoustic filters requires many trial-and-error iterations over the shape. Furthermore, the design space is often limited to simple geometries such as pipes (e.g., for making flutes, trumpets, and industrial

Figure 4.1: **Acoustic Tagging.** By optimizing the structure of primitives **(a)**, we control the acoustic response of an object when it is tapped **(c)** and thereby tag the object acoustically. Given three objects with identical shapes **(b)**, we can use a smartphone to read the acoustic tags in realtime, by recording and analyzing the tapping sound, and thereby identify each object.

mufflers) since the acoustic behavior of only these simple shapes can be easily characterized. Current computational design tools support only these simple primitives and even then the design process requires strong expertise in this domain.

Meanwhile, recent advances in additive manufacturing have significantly facilitated rapid manufacturing of complex geometries. This trend opens up new possibilities for expanding the design space of acoustic filters, thus motivating the development of corresponding computational methods that can efficiently simulate and optimize the shape of the cavity in order to achieve desired acoustic filtering effects. In light of this, the goal of our work is to expand the range of acoustic filter design by employing complex cavity shapes computationally optimized and then physically realized using additive manufacturing.

We propose *Acoustic Voxels*, a computational method that assembles basic shape primitives into a complex geometry, one that produces the desired acoustic filtering. In particular, we consider a simple type of shape primitive, a hollow cube with circular holes on some of its six faces (Figure 4.4). We show that these primitives, albeit simple individually, offer a large design space for acoustic filters when *modularly* joined at their faces into a complex assembly. This modular scheme also permits fast and accurate estimation of the acoustic performance of a given assembly, thereby allowing automatically optimizing its structure to achieve target acoustic filtering properties while

satisfying geometric constraints of inlet/outlet positions and overall shapes.

Our approach starts with precomputing the acoustic transmission for our parameterized shape primitives. At runtime, given an arbitrary assembly of these primitive filters, our method estimates its acoustic transmission, predicting the boosted and suppressed frequency regions. This, in turn, enables us to derive a formula to compute the gradient of the acoustic transfer with respect to shape parameters, and further develop an efficient combinatorial and continuous optimization algorithm to design desired filter structures. Our method combines a stochastic optimization method for computing the topology of the assembly (i.e., the way of arranging and connecting the primitives) with a gradient-based quasi-Newton method for computing the geometric parameters of each primitive shape in the assembly. We validate our method by running finite-element off-line acoustic simulation and industrial laboratory tests performed by acoustic engineering professionals (§4.5).

Our proposed approach automates the design of acoustic filters. This simplified design process allows casual users to produce objects with custom acoustic properties. Our method also expands the range of acoustic filters that can be achieved, enabling exploration of many different applications. In addition to designing different types of noise attenuation components (e.g., mufflers), our method can customize musical instruments with non-conventional shapes. Furthermore, we can embed imperceptible acoustic information into the fabricated objects, and thus opens up new types of interactions with fabricated objects, extending current visually based design into audiovisual design.

4.2 Background on Acoustic Filters

We start by briefly reviewing the theory of acoustic filters and refer to the textbooks [Ingard, 2009; Munjal, 2014] for more details. A typical acoustic filter has a cavity structure connecting an inlet and an outlet— trumpets and motorcycle mufflers are classic examples. When sound waves enter into

the inlet, travel through the cavity, and leave from the outlet, their frequency components are altered. In most applications, the physical size of a filter ranges from centimeters to tens of centimeters and their operating frequencies are up to thousands of Hz. To evaluate the performance of acoustic filters, the following two quantities are often used (Figure 4.2):

- **Input impedance.** Consider a steady-state sound transmission through a filter. In the frequency domain, the sound pressure and acoustic velocity at a location $\boldsymbol{x}$ are denoted as $p(\boldsymbol{x}, \omega)$ and $v(\boldsymbol{x}, \omega)$, respectively. The *acoustic impedance*, defined as $Z(\boldsymbol{x}, \omega) = \frac{p(\boldsymbol{x},\omega)}{v(\boldsymbol{x},\omega)}$, indicates how much sound pressure is generated by a given air vibration of frequency ω at position $\boldsymbol{x}$. Particularly, we are interested in the impedance value at the inlet $\boldsymbol{x}_i$, $Z_{\text{IN}}(\omega) = Z(\boldsymbol{x}_i, \omega)$, called *input impedance* (Fig. 4.2). $Z_{\text{IN}}(\omega)$ usually varies strongly with respect to the frequency and has multiple local minima and maxima, which correspond to the sound frequencies that are the easiest and the most difficult to transmit through the filter. For example, the playing frequencies of a trumpet are very close to the local maxima of its input impedance.

- **Transmission loss.** To design acoustic filters for noise reduction, a widely used measure is the *transmission loss* [Munjal, 2014], defined as the ratio, expressed in decibels (dB), of the acoustic power incident to the muffler to the power transmitted downstream into the environment. Concretely, if the inlet and the outlet of an acoustic filter are sufficiently small, its transmission loss is described as:

$$L_{\text{TL}}(\omega) = 10 \log_{10} \left| \frac{S_{\text{i}} p_{\text{i+}}^2(\omega)}{S_{\text{o}} p_{\text{o}}^2(\omega)} \right|,$$

  where S_{i} and S_{o} are the cross-sectional area of the inlet and the outlet, respectively; $p_{\text{o}}(\omega)$ is the frequency-domain acoustic pressure of the transmitted sound wave at the outlet, away from the filter; and $p_{\text{i+}}(\omega)$ is the acoustic pressure of the incident wave at the inlet, also in the frequency domain. In short, $L_{\text{TL}}(\omega)$ measures how much the sound wave of frequency ω gets

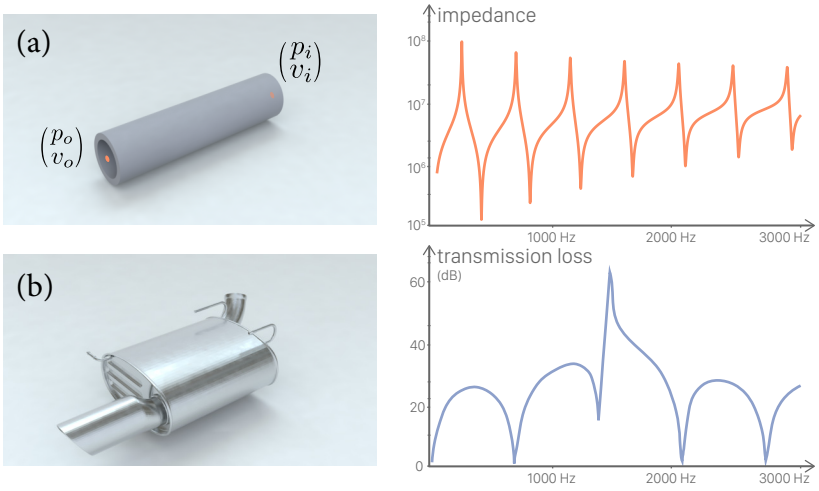


Figure 4.2: **Acoustic filters examples.** (a) A duct as part of a wood instrument is measured using the input acoustic impedance; (b) Mufflers are often evaluated using transmission loss.

attenuated when passing through the filter.

Depending on specific applications, our goal is to optimize the internal structure of a filter in order to obtain target input impedance or transmission loss in a frequency range. In general, accurately predicting these quantities requires solving the acoustic wave equation or, in the frequency domain, the Helmholtz equation [Pierce and others, 1991; Allen and Raghuvanshi, 2015]. Either approach is computationally expensive, especially for complex filter structures. Notably, the relationship between the geometry of the filter and the resulting impedance or transmission loss function is rather complex, obstructing us from formulating a well-defined optimization problem of this geometry. Therefore, we take a different approach by leveraging the concept of the transmission matrix.

Transmission matrix If both the inlet and the outlet have a small cross-section, much smaller than the wavelength of the operating sound waves, one can reasonably assume that the acoustic pressure and velocity are both distributed uniformly over the cross-section [Ingard, 2009; Rienstra and Hirschberg, 2003]. In our examples, the highest frequency that we modeled is 4500Hz, having a wavelength around 7.6cm. And our cross section radius ranges from 3mm to 1cm. We vali-

dated this assumption with industrial lab measurements (§4.5.1) and physical fabrication (§4.5.2-§4.5.4).

Let $(p_{\mathrm{i}}(\omega), v_{\mathrm{i}}(\omega))$ and $(p_{\mathrm{o}}(\omega), v_{\mathrm{o}}(\omega))$ denote the complex-valued acoustic pressure and velocity in frequency domain, at the cross-sections of the inlet and the outlet, respectively (Figure 4.2-a). Their relationship can be approximated linearly,

$$\begin{pmatrix} p_{\mathrm{o}}(\omega) \\ p_{\mathrm{i}}(\omega) \end{pmatrix} = \begin{pmatrix} T_{11}^{\omega} & T_{12}^{\omega} \\ T_{21}^{\omega} & T_{22}^{\omega} \end{pmatrix} \begin{pmatrix} v_{\mathrm{o}}(\omega) \\ v_{\mathrm{i}}(\omega) \end{pmatrix}, \tag{4.1}$$

where T_{ij}^{ω} is i-th row and j-th column in the complex-valued *transmission matrix* at frequency ω. In this thesis, we also denote this matrix as $\mathsf{T}(\omega)$ to emphasize its frequency dependence. Transmission matrices have been widely used in industrial muffler design [Ingard, 2009], as it relates to the input impedance and transmission loss through simple formulas:

$$Z_{\mathrm{IN}}(\omega) = \frac{T_{11}^{\omega} + T_{12}^{\omega}T_{21}^{\omega} - T_{11}^{\omega}T_{22}^{\omega}}{1 - T_{22}^{\omega}} \text{ and} \tag{4.2}$$

$$L_{\mathrm{TL}}(\omega) = 20\log_{10}\left(\frac{1}{2}\left|\frac{T_{11}^{\omega}}{T_{21}^{\omega}} + \frac{T_{12}^{\omega}}{\rho c} - \frac{T_{11}^{\omega}T_{22}^{\omega}}{\rho c T_{21}^{\omega}} + \frac{\rho c}{T_{21}^{\omega}} - \frac{T_{22}^{\omega}}{T_{21}^{\omega}}\right|\right) \tag{4.3}$$

where ρ is the air density and c is the sound speed.

Challenges Unfortunately, computing the transmission matrix of a filter structure is expensive. For each frequency ω, the standard approach first samples two sets of pressures, $(p_{\mathrm{i}1}, p_{\mathrm{o}1})$ and $(p_{\mathrm{i}2}, p_{\mathrm{o}2})$, at the inlet and the outlet. Each set of pressures, together with the zero-normal-velocity condition on the solid boundary of the filter, forms a complete boundary condition that can be used to solve the Helmholtz equation and uniquely determine the acoustic velocity $(v_{\mathrm{i}1}, v_{\mathrm{o}1})$ (and $(v_{\mathrm{i}2}, v_{\mathrm{o}2})$) at the inlet and outlet. Then, the transmission matrix can be computed by solving a 2×2

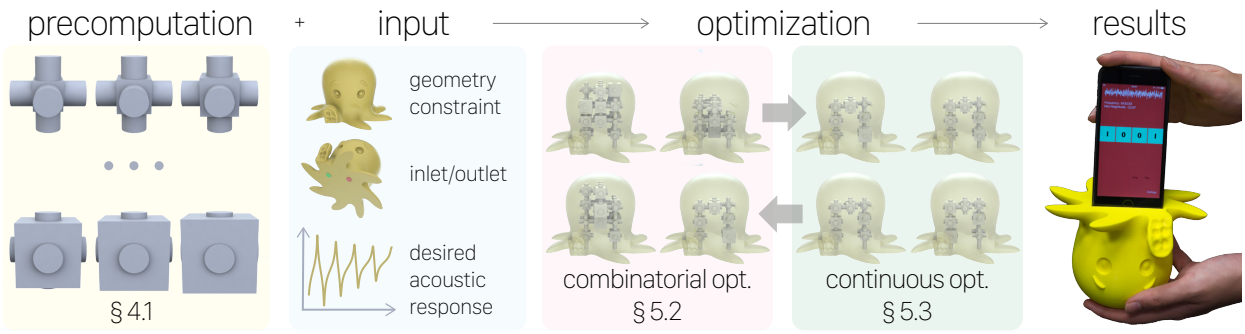


Figure 4.3: **Overview.** Our method exploits precomputed transmission matrices of the primitives and uses a combinatorial and continuous optimization to construct the assembly of filters. Please refer to §4.2.1 for an outline of each step.

linear system,

$$\begin{pmatrix} p_{o1}(\omega) & p_{o2}(\omega) \\ p_{i1}(\omega) & p_{i2}(\omega) \end{pmatrix} = \begin{pmatrix} T^{\omega}_{11} & T^{\omega}_{12} \\ T^{\omega}_{21} & T^{\omega}_{22} \end{pmatrix} \begin{pmatrix} v_{o1}(\omega) & v_{o2}(\omega) \\ v_{i1}(\omega) & v_{i2}(\omega) \end{pmatrix}. \tag{4.4}$$

This process needs to solve the Helmholtz equation twice for each frequency ω. In addition, while it is straightforward to compute impedance and transmission loss using the transmission matrix, it remains hard, if not impossible, to compute the gradient of the transmission matrix with respect to the geometric parameters of the filter—this gradient is needed for optimizing the cavity shape of the filter (§4.4). Our approach addresses all these challenges.

4.2.1 Method Overview

Our method automatically constructs the internal structure of an acoustic filter that connects an inlet and an outlet of a 3D volume (Figure 4.4-right). We aim to control its input impedance or the transmission loss function as specified by the user.

System input and output Concretely, out method takes as input three components, (i) a 3D volume in which the acoustic filter is placed, (ii) the positions of the inlet and the outlet, specified on the surface of the 3D volume, and (iii) the frequency locations to be boosted or suppressed. It then outputs the transmission geometry that fits into the 3D volume and that can be fabricated to produce the desired filtering effects (Figure 4.3).

To this end, we propose a primitive acoustic resonator, a family of simple hollow shapes serving as building blocks to assemble a complex acoustic filter. These primitives allow us to precompute their transmission matrices, which in turn enable a fast runtime algorithm to compute the acoustic impedance and transmission loss of any filters made from an assembly of the primitive resonators (§4.3). Leveraging the fast computation of transmission matrices, we further address the optimization of the inverse problem (§4.4), one that finds an assembly of the primitive shapes to achieve a target acoustic input impedance or transmission loss. To this effect, we formulate a combinatorial and continuous optimization problem, combinatorial in the sense of how to connect primitive shapes, and continuous in the sense of determining geometric parameters of the primitives. To solve it, we propose a hybrid method that interleaves a stochastic optimization, namely the Sequential Monte Carlo method, with a gradient-based quasi-Newton scheme.

4.3 Modular Acoustic Filter

4.3.1 Primitive Resonator

We propose to use a simple shape as our primitive resonator—a hollow cube with extruded cylinders on its six faces (Figure 4.4-left). All the cylindrical extrusions have the same radius and length and therefore the bounding boxes of all primitives stay the same. They can be composed together at their faces by connecting an inlet and an outlet to form a complex structure (Figure 4.4-right).

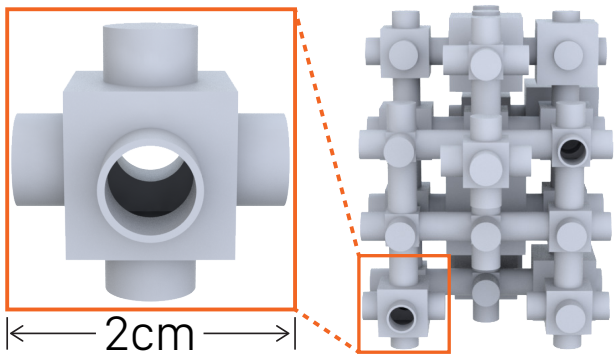


Figure 4.4: **Modular filter.** Our primitive resonator is a single shape bounded by a 2cm×2cm×2cm cube (left). A combination of the primitives with varying shape parameters can form complex structure that connects an inlet to an outlet.

Furthermore, the size of each hollow cube can change, providing one degree of freedom per element as control variables to influence acoustic filtering properties, in addition to the connectivity of the resonators.

Rationale Using the simple primitives offers many advantages: (i) They can fill the interior volume of virtually any shape, as long as they are sufficiently small. This enables us to construct acoustic filters subject to various shape constraints. (ii) Computing the transmission matrix of any assembly becomes fast and accurate. (iii) With a hollow cube of a variable size, the primitive is in a one-dimensional shape space, which can be easily sample. For each sample, we precompute its transmission matrices and interpolate between neighboring transmission matrices. When composed into an assembly, these primitives offer a large number of degrees of freedom for controlling acoustic filtering properties. This idea is also similar to the concept of 3D symmetric condensed nodes for the computation of electromagnetic fields [Christopoulos, 2006].

In this section, we describe how we compute the transmission matrix of an arbitrary assembly of the primitive resonators. Ultimately, our goal is to compute both the *topology* of the primitive assembly and the *geometric parameters* (i.e., the cube size of each element) for a desired input impedance or

transmission loss.

Multi-port transmission matrix We start by extending the concept of transmission matrix in (4.1) into a six-port transmission matrix. Since the radii of the six open ports of a primitive shape are small, it remains valid to assume that the frequency-domain acoustic pressure $p_{\mathrm{i}}(\omega)$ and velocity $v_{\mathrm{i}}(\omega)$ (i=1...6) are uniformly distributed over the cross sections at each port. Then a linear relationship similar to (4.1) holds:

$$\begin{pmatrix} p_1(\omega) \\ \vdots \\ p_6(\omega) \end{pmatrix} = \begin{pmatrix} T^{\omega}_{11} & \dots & T^{\omega}_{16} \\ \vdots & \ddots & \vdots \\ T^{\omega}_{61} & \dots & T^{\omega}_{66} \end{pmatrix} \begin{pmatrix} v_1(\omega) \\ \vdots \\ v_6(\omega) \end{pmatrix}. \tag{4.5}$$

In a way similar to Eq. (4.4), we compute for a given frequency ω the six-port transmission matrix T by sampling six different sets of pressures $\{p_i, i = 1...6\}$. Each set of pressures establishes the (Dirichlet) boundary condition that uniquely solves the Helmholtz equation for sound propagation in the primitive resonator. After the six Helmholtz solves, it produces six corresponding acoustic velocities $\{v_i, i = 1...6\}$, which, together with $\{p_i\}$, can be substituted into Eq. (4.5) and uniquely determine the matrix T.

Precomputation Given a primitive resonator shape with six ports, precomputing the transmission matrix T^{ω} amounts to solving the Helmholtz equation 6 times, with different Neumann boundary conditions. In particular, for the i-th solve, we set $v_i = 1$ and $v_j = 0$ for all $j \neq i$. The transmission matrix is calculated as $T^{\omega}_{ji} = p^i_j$, where p^i_j is the solution on face j under the i-th boundary condition. This transmission matrix depends on not only the frequency but also the shape parameter, the cube size. Therefore, we sample a set of frequency values and cube sizes, precompute the T matrices, and store them in a database. We also note that the precomputation step can be accelerated

using the recent asymptotic frequency sweeping method [Li *et al.*, 2015]. With these precomputed six-port transmission matrices, we are able to interpolate the matrix of a primitive resonator of any frequency and cube size in the sampled range. This interpolation will be used later in the optimization step (§4.4) to compute optimal topology and geometry of the primitive assembly for a target acoustic filtering property.

4.3.2 Transmission Matrix of Resonator Assembly

Now we compute the transmission matrix of a resonator assembly, which consists of primitive resonators (the size of each resonator is specified). Each of the six ports of a resonator is either joined with a port of another resonator or closed with a solid wall. These ports are connected (possibly through multiple paths) from an inlet to an outlet. Our goal here is to compute the frequency-dependent 2×2 transmission matrix that relates the acoustic pressure and velocity at the outlet to those at the inlet, as described in (4.1).

We start with some notation. Consider an assembly composed of N primitive resonators. We use j to index the primitives and k to index the six ports of each primitive. Let $p_k^j(\omega)$ and $v_k^j(\omega)$ denote respectively the frequency-domain acoustic pressure and velocity at the k-th port of the j-th primitive. From the precomputation, for each primitive resonator j we also have a six-port (6×6) transmission matrix $\mathsf{T}_j(\omega)$ that relates the pressures $p_k^j(\omega)$ with velocities $v_k^j(\omega), k = 1...6$ at its six ports.

Similarly to the method used in Eq. (4.4), we sample two sets of pressures, $(\bar{p}_{\text{i1}}, \bar{p}_{\text{o1}})$ and $(\bar{p}_{\text{i2}}, \bar{p}_{\text{o2}})$, at the inlet and the outlet. We seek a fast method to compute the corresponding acoustic velocity, $(\bar{v}_{\text{i1}}, \bar{v}_{\text{o1}})$ and $(\bar{v}_{\text{i2}}, \bar{v}_{\text{o2}})$, without solving the expensive Helmholtz equations. We observe that we can

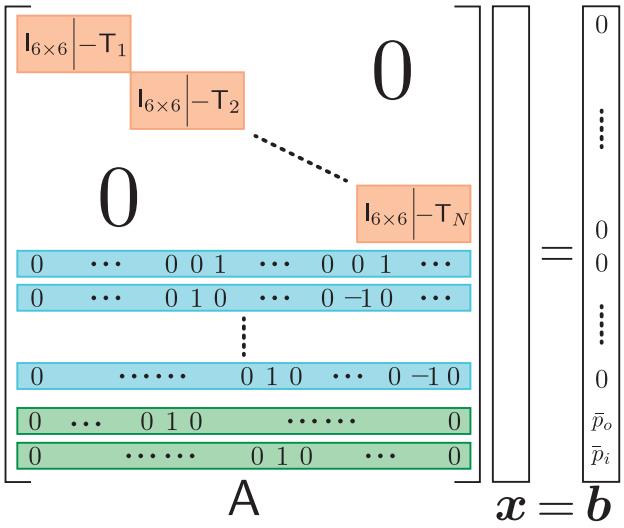


Figure 4.5: **Linear solve of a filter assembly.** The top orange part refers to the transmission matrices related to each node in the assembly. The middle blue part specifies the connection information by mapping the velocity and pressure values. The bottom two green rows are the given boundary conditions at the inlet and the outlet.

construct a sparse linear system (visualized in Figure 4.5),

$$\mathsf{A}(\omega)\boldsymbol{x}(\omega) = \boldsymbol{b}(\omega), \tag{4.6}$$

to solve for the pressures $p_k^j(\omega)$ and velocities $v_k^j(\omega)$ of all ports ($j = 1...N, k = 1...6$). Here, $\boldsymbol{x}$ has $12N$ elements, stacking all the pressures and velocities of frequency ω at all ports. Every resonator contributes a linear relationship (4.5), resulting in a 6 linear equations which appears as a 6×12 submatrix (orange blocks in Figure 4.5). All the resonators together form a $6N \times 12N$ sub-block matrix. In addition, for the two ports that connect to the inlet and outlet, the pressures are the sampled values (i.e., the two green rows in Figure 4.5); at the closed ports, the velocities vanish; at every pair of connected ports, their pressures need to match and their velocities need to be additively inverse (e.g., the blue rows in Figure 4.5), as the sound waves flow along the same direction. All these constraints result in another $6N$ linear equations. Putting together these equations yield a full-rank sparse and $12N \times 12N$ linear system.

We also note that the matrix A depends on the cube sizes of the primitive resonators, as it is assembled using their transmission matrices T_i, but $\boldsymbol{b}$ is a constant. Later when optimizing the cube sizes, we will compute the derivative of A with respect to each cube size.

Computational efficiency This process computes the transmission matrix at a frequency ω by solving the sparse linear system, $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$, twice. Both have the same A matrix, so it only needs to be factorized once. In addition, across all frequencies, the sparsity pattern of A stays the same. To exploit this invariant, we use the symbolic factorization (reordering) only once for the entire computation and update the numerical data for each frequency sample, all implemented using the Direct Sparse Solver provided in Intel MKL. As a result, the computation of transmission matrices for all frequency samples (nearly 1000 samples) typically finishes in a few seconds.

4.4 Optimization

We now focus on the inverse problem: computing a structure of a primitive assembly and the parameter of each primitive in the assembly in order to realize a desired acoustic filtering property. We formulate this problem as a combinatorial and continuous optimization (§4.4.1). To address both the combinatorial and the continuous aspects of the problem, our algorithm interleaves a stochastic optimization method with a quasi-Newton method (§4.4.2 and §4.4.3).

4.4.1 Problem Formulation

Optimization objective Our optimization goal, the acoustic filtering property, depends on a specific application, whether it is a target impedance $Z_{\text{IN}}(\omega)$ (e.g., for wind instruments) or a target transmission loss $L_{\text{TL}}(\omega)$ (e.g., for engine mufflers) in a frequency range $[\omega_l, \omega_r]$. Both quantities can

be computed from the transmission matrix $\mathsf{T}(\omega)$ of a given assembly using Eq. (4.2) and Eq. (4.3), respectively. Thus, we discretize the frequency range using a set of samples $\omega_i \in [\omega_l, \omega_r], i = 1...N_\omega$ and define a unified objective function in a least-squares form:

$$J = \sum_{i=1}^{N_\omega} \left(g(\mathsf{T}(\omega_i)) - \bar{g}_i \right)^2 . \tag{4.7}$$

Here, $g(\mathsf{T}(\omega_i))$ is the acoustic filtering quantity depending on the transmission matrix at a sampled frequency ω_i. For instance, to control the input impedance, we use $g(\mathsf{T}(\omega_i)) = \log_{10} |Z_{\text{IN}}(\omega_i)|$; to control the transmission loss, we use $g(\mathsf{T}(\omega_i)) = L_{\text{TL}}(\omega_i)$. $\bar{g}_i$ is the target acoustic filtering quantity at the frequency ω_i. These values are user-controlled, e.g., by specifying a target curve in the frequency domain.

We note that while this objective function suits well for our applications (§4.5), our optimization method does not depend on this particular choice, as presented in the rest of this section.

Shape constraint In many applications, filters are often embedded in a limited space. To account for this requirement, we allow the user to specify a 3D surface mesh to constrain the volume of the assembly in the optimization process. Before the optimization starts, we voxelize the 3D mesh into a lattice, where each grid cell represents a possible placement of a primitive resonator, and the grid connects an inlet and an outlet, both specified on the mesh boundary (see video). By construction, the resulting assembly of resonators are guaranteed to satisfy the shape constraint and connect the inlet and outlet.

Optimization variables We have two types of optimization variables: (i) a string of binary bits $\boldsymbol{s}$ indicating the lattice grid connectivity and (ii) a vector $\boldsymbol{u}$ stacking the cube sizes of primitive resonators used in the assembly. We index each grid cell interface in the lattice. If two primitives are joined at an interface i, then the corresponding bit in $\boldsymbol{s}$ is set to one. If a face of the grid cell is not con-

nected with its neighboring grid cell, the corresponding bit in $\boldsymbol{s}$ is set to zero and the resonator port on that face is closed with a solid boundary. As we will describe later, this bit string representation is particularly suitable for our stochastic sampling algorithm. With these optimization variables, we rewrite the acoustic filtering quantity $g(\mathsf{T}(\omega))$ in Eq. (4.7) as $g(\mathsf{T}(\boldsymbol{s}, \boldsymbol{u}, \omega_i))$ and explicitly write J as $J(\boldsymbol{s}, \boldsymbol{u})$ because the transmission matrix T depends on both the topology (described by $\boldsymbol{s}$) and the geometry (described by $\boldsymbol{u}$) of the primitive assembly.

Method rationale and overview The optimization variables reflect the combinatorial and continuous nature of our problem. The problem of determining the placement and connectivity of the primitives in the lattice is combinatorial; and determining the cube sizes of each primitive is continuous. A typical method of solving a combinatorial optimization relies on a Monte Carlo method to sample in the parameter space and accept or reject samples probabilistically. The efficiency of this method critically depends on the performance of evaluating the objective function, as it often requires a large number of samples. From this perspective, our fast computation of the transmission matrix (§4.3.2), a necessary component for evaluating the objective function (4.7), lays out an important cornerstone for using a stochastic optimization algorithm. Meanwhile, if the connectivity is given, optimizing the cube sizes for each primitive is a continuous problem, for which a gradient-based method is more efficient.

We propose to use a stochastic optimization method to optimize the connectivity of the primitives. When evaluating the objective function of a sampled resonator structure (i.e., the $\boldsymbol{s}$), we compute the cube size for each primitive (i.e., the $\boldsymbol{u}$) using a gradient-based continuous optimization method that minimizes the objective function with the fixed resonator structure. This is because continuous optimization, leveraging gradient descent, is more efficient than stochastically sampling cube sizes. Effectively, our method is a hybrid that interleaves a Monte Carlo sampling with a quasi-Newton optimization scheme.

4.4.2 Combinatorial Optimization of Connectivity

To solve a combinatorial optimization problem, one simple and popular approach is to use simulated annealing [Kirkpatrick *et al.*, 1983], a method that can be interpreted as a single sequence of Markov-Chain Monte Carlo (MCMC) sampling [Robert and Casella, 2013]. One way of improving its efficiency is to use multiple sequences of MCMC sampling, for which an efficient method is Sequential Monte Carlo (SMC). In computer graphics, SMC has been applied for rendering, character control, and procedural modeling [Pegoraro *et al.*, 2008; Hämäläinen *et al.*, 2014; Ritchie *et al.*, 2015]. In numerical optimization, SMC methods have been used for optimizing non-convex, non-differentiable, and high-dimensional objective functions [Mıguez *et al.*, 2010]. In the following, we outline our modified SMC algorithm, followed by highlighting the components that are specifically tailored for our problem.

Modified SMC algorithm As outlined in Algorithm 3, we maintain N_s different samples of the lattice connectivity, that is, a set of binary-bit strings $\{\boldsymbol{s}_i, i = 1...N_s\}$. At each iteration, the algorithm performs the following steps:

1. Evaluate the objective function J_i for each sampled connectivity $\boldsymbol{s}_i$, $i = 1...N_s$ (Line 3-6 in Algorithm 3).

2. Select the best M samples that produce the lowest objective values and perturb them. The perturbation of bit strings is similar to the mutation operation in a genetic algorithm.

3. Replace the rest of the $N_s - M$ samples with new samples using an MCMC sampling step (Line 13-14 in Algorithm 3).

These steps repeat until the best objective value drops below a threshold (Line 7 in Algorithm 3).

Evaluation of objective function Given a sampled connectivity, we evaluate the objective function J defined in §4.4.1. Since J depends on both the connectivity and the primitive cube sizes and the latter has not yet been determined, we treat the evaluation as another optimization problem, one that minimize the objective function over all possible cube sizes but with a fixed connectivity. This is a continuous optimization problem, which we solve in §4.4.3.

Random sample of connectivity To initialize the set of lattice connectivities and to replace the worst $N_s - M$ samples at the third step of the algorithm, we need to sample bit strings $\boldsymbol{s}_i$. To this end, we use a simple rejection sampling scheme, starting by random sampling of a bit string. Since we must ensure the inlet and outlet are connected through primitive resonators, after sampling a bit string we verify whether the corresponding connectivity structure connects the inlet with the outlet (e.g., using a depth-first search on the lattice) and reject the sample if does not.

Connectivity perturbation We perturb the connectivity string $\boldsymbol{s}_i$ using a mutation. Specifically, we randomly select a bit in a string s_i and flip it. In addition, this mutated string is subject to two constraints: (i) the corresponding connectivity structure needs to retain the connection between the inlet and the outlet; and (ii) the mutated bit needs to influence the resonator paths that connect the inlet and the outlet; otherwise, the mutation makes no difference to the connected component between the inlet and the outlet. We check the mutated bit string against both requirements and reject the mutation if it fails the check.

4.4.3 Local Continuous Optimization

Next we discuss how to evaluate the objective function after sampling a lattice structure. This evaluation optimizes the cube sizes $\boldsymbol{u}$ of each primitive in the lattice structure in order to compute the minimal objective function value. To achieve this, we first compute the gradient of J with respect

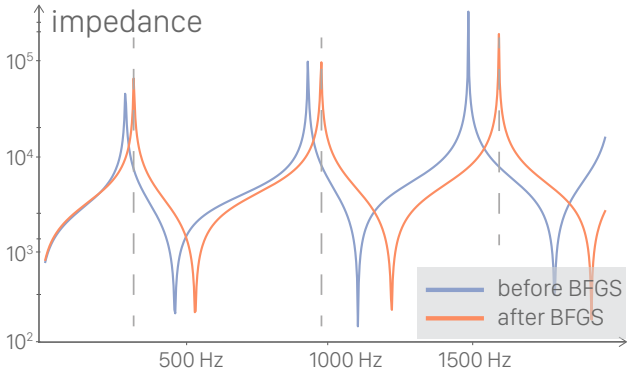


Figure 4.6: Before and after Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization. Combinatorial sampling is difficult to converge to the user-specified target quickly due to its random nature. Enforcing local optimization for each sample reaches the desired acoustic target faster.

to $\boldsymbol{u}$ from Eq. (4.7),

$$\frac{\partial J(\boldsymbol{s},\boldsymbol{u})}{\partial \boldsymbol{u}} = 2\sum_{i=1}^{N_\omega}(g(\mathsf{T}(\boldsymbol{s},\boldsymbol{u},\omega_i)) - \bar{g}_i)\frac{\partial g(\mathsf{T}(\boldsymbol{s},\boldsymbol{u},\omega_i))}{\partial \boldsymbol{u}}. \tag{4.8}$$

The function g depends on the transmission matrix T, which further depends on the acoustic pressures and velocities at every port of all the primitive resonators, according to Eq. (4.6). To compute the partial derivative of g, applying the chain rule yields:

$$\frac{\partial g(\mathsf{T}(\boldsymbol{s},\boldsymbol{u},\omega_i))}{\partial \boldsymbol{u}} = \underbrace{\left(\frac{\partial g}{\partial \mathsf{T}}\frac{\partial \mathsf{T}}{\partial \boldsymbol{x}}\right)}_{\boldsymbol{m}^T}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}}, \tag{4.9}$$

where $\boldsymbol{x}$, as used in Eq. (4.6), stacks frequency-domain pressures and velocities of all the ports of the primitives. If N denotes the number of primitive resonators of the assembly, then $\boldsymbol{m}$ is a vector of the length $12N$, independent of the cube sizes of the primitives. $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}}$ is a $12N \times N$ matrix. To compute this matrix, recall that in Eq. (4.6), the matrix A depends on the cube sizes of the primitives, and $\boldsymbol{b}$ is a constant. Differentiating both sides of Eq. (4.6) with respect to $\boldsymbol{u}$ yields:

$$\mathsf{A}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} + \frac{\partial \mathsf{A}}{\partial \boldsymbol{u}} = \boldsymbol{0}, \tag{4.10}$$

which is a linear system with N right-hand-side vectors, $-\frac{\partial \mathsf{A}}{\partial \boldsymbol{u}}$. Since A is assembled from the transmission matrices of all primitives in the assembly(recall §4.3.2), $\frac{\partial \mathsf{A}}{\partial \boldsymbol{u}}$ involves the derivatives of the transmission matrices with respect to the cube sizes. We compute them by interpolating our precomputed primitive transmission matrices.

It seems straightforward to compute $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}}$ by factorizing A only once and solving the linear system N times, and use Eq. (4.9) and (4.8) to compute the gradient of the objective function. However, if N is large, even the repeated back substitutions for solving Eq. (4.10) are slow. Especially when used in a Monte Carlo sampling step, this would significantly reduce the efficiency of the overall optimization algorithm.

Speedup with Adjoint Method Fortunately, this computation can be largely accelerated using the adjoint method, one that has been applied in computer graphics mainly for animation control problems [McNamara *et al.*, 2004; Wojtan *et al.*, 2006; Barbič *et al.*, 2009]. The key idea is based on the observation that computing a matrix-vector product, $\boldsymbol{m}^T\mathsf{B}$ such that $\mathsf{AB} = \mathsf{C}$, is equivalent to computing $\boldsymbol{t}^T\mathsf{C}$ such that $\mathsf{A}^T\boldsymbol{t} = \boldsymbol{m}$. The advantage of the latter is that only a single linear-system solve for the vector $\boldsymbol{t}$ is needed. In our problem, this amounts to first solving

$$\mathsf{A}^T\boldsymbol{t} = \left(\frac{\partial g}{\partial \mathsf{T}}\frac{\partial \mathsf{T}}{\partial \boldsymbol{x}}\right), \text{ followed by computing } \frac{\partial g}{\partial \boldsymbol{u}} = \boldsymbol{t}^T\frac{\partial \mathsf{A}}{\partial \boldsymbol{u}}. \tag{4.11}$$

For all our examples, this method results in nearly $10\times$ speedups over the straightforward approach.

With the computation of the gradient $\frac{\partial J}{\partial \boldsymbol{u}}$ depicted, we apply it to a quasi-Newton method to minimize J. In our implementation, we use the Limited-memory BFGS Bounded (L-BFGS-B) [Zhu *et al.*, 1997a]. In practice we found local gradient descent step complements the combinatorial sampling. Figure 4.6 illustrates the effectiveness of the local optimization of the impedance curve.

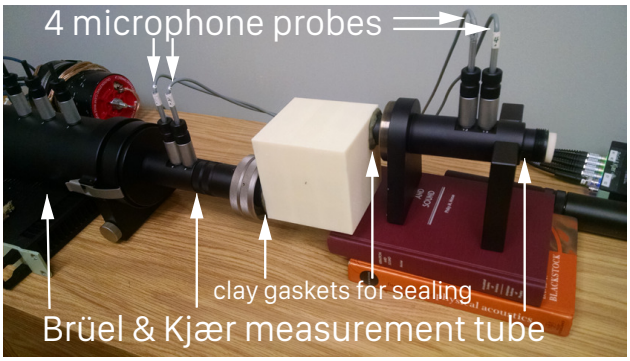


Figure 4.7: Industrial laboratory measurement setup.

4.5 Results

We now present the experiments we conducted to test our method. In all examples, we sample the frequency range every 3Hz from 20Hz to 5kHz to precompute transmission matrices. The cube size of the primitive resonator varies depending on specific applications: For muffler design and acoustic signatures, the cube size is between 6mm and 2cm, sampled every 1mm. For laboratory tests and wind instrument design, the cube size is between 25cm and 35mm, also sampled every 1mm. The precomputation takes a few hours on a 16-core cluster.

We fabricated our designs using Stratasys uPrint SE Plus, a filament-based 3D printer with a layer resolution at 0.254mm. We use ABS-P430 plastic as the model material and a dissolvable support material which can be washed away upon finish. The fabrication time varies from a few hours to a day, primarily depending on geometric size of a given model.

4.5.1 Validation on Acoustic Voxels

The fundamental building block of our assembly structure optimization is the fast computation of a transmission matrix for an assembly (recall §4.3.2). We validate its accuracy using finite-element-method (FEM) simulation and industrial laboratory tests.

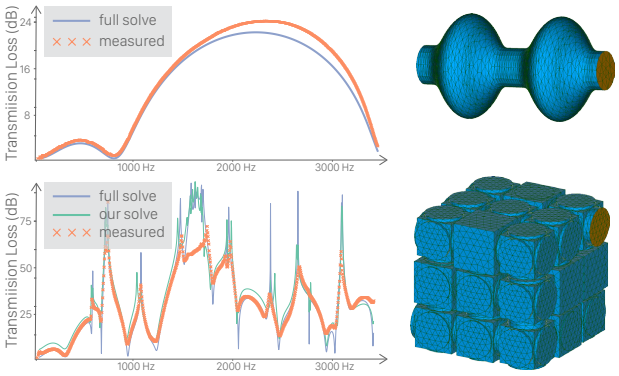


Figure 4.8: Double muffler and cube measured by Brüel & Kjær. Our method agrees closely with both the expensive FEM solve and the lab measurement. There is a large difference around 1600Hz, where the measurement input signal does not have sufficient power to pass through. We believe this is caused by the wide and high transmission loss values around this region which lead to low signal-to-noise ratio (SNR) during measurement.

Finite-element simulation We compute the transmission loss using `Code_Aster` [Aubry, 2013], a well-developed and carefully tested finite-element solver for mechanics. We follow the routine outlined in §5.2.2, solving for the acoustic velocities with different boundary conditions using `Code_Aster`. We then compute the transmission matrix by assembling and solving the Equation (4.4).

Industrial laboratory test We sent fabricated samples to Brüel & Kjær's acoustic laboratory for independent, third-party tests conducted by their acoustic professionals. Brüel & Kjær is the world's largest manufacturer and supplier of acoustic measurement equipment and solutions. They measured the transmission loss of our samples using Brüel & Kjær4206-T measurement tubes with the 4-microphone technique [Tao and Seybert, 2003], sweeping the frequency range every 4Hz from 20Hz to 3500Hz under the condition of 21°C (room temperature), 98.9kPa (pressure), and 44% of relative humidity. To ensure best acoustic seal during the tests, clay gaskets were also added between the measurement tubes and our test samples (see Fig. 4.7).

Comparison The comparison shows that our fast computation of transmission loss agrees with both the finite-element simulation and laboratory experiments closely, as in Figure 4.8. The top

plot in Figure 4.8 validates the agreement between the finite-element simulation and the laboratory tests using a double-chamber muffler, which is known to be an effective broadband filter. It lacks the curve from our computation model, simply because this model is not made from our primitive resonators. We use this test to examine the use of the numerical and experimental methods. The bottom plot reports the transmission loss of an assembly muffler made of $3\times3\times3$ primitive resonators, comparing the results from finite-element simulation (blue curve), Brüel & Kjær's laboratory measurement (orange dots), and our fast computation (blue curve). They all agree with each other closely. Particularly, our computational model is able to predict the peaks and valleys on the transmission loss curve, with the differences from the measurement less than 20Hz on average. These peaks and valleys indicate the most and the least attenuated frequencies when sound passes through the filter, and they will be of practical importance to control when one designs a muffler, as demonstrated later in §4.5.2.

We also run three validation tests on impedance curves to compare the error of our fast computation and full FEM solve. Figure 4.9 shows that our method robustly computes the impedance curve and introduces slight numerial instability as the model gets more complicated. In terms of computational performance, our method is much faster than the finite-element simulation. For example, to compute the transmission loss curve of this $3\times3\times3$ resonator assembly, which involves compu-

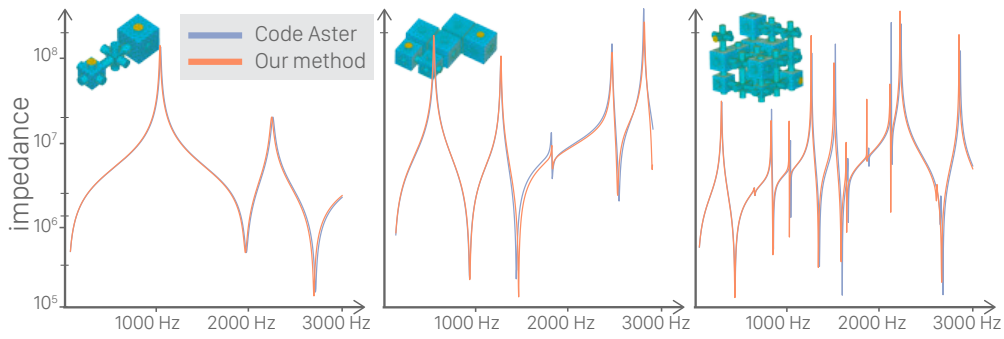


Figure 4.9: **Impedance comparison** with Code Aster. In the sequence of three models with increasing complexity, our method agrees with Code Aster closely.

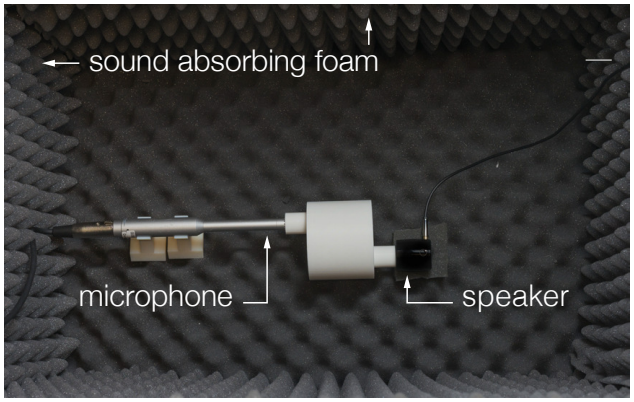


Figure 4.10: **Recording setup** to record the sounds before and after our filtering. The chamber inner surface is surrounded by sound absorbing foam to minimize ambient noise from outside as well as the wave reflection/refraction inside the chamber.

tation at 1000 frequency samples, our method takes 1.2 seconds, while the finite-element method takes around 22 hours, resulting in 77,000$\times$ speedup.

4.5.2 Application I: Muffler Design

Man-made mechanisms produce noise, with clear patterns exhibited in the sound spectrum. For instance, the aircraft and automobile engine noise have pronounced frequency components related to revolutions per minute (RPM) of the engine cranks. The car horns have particular frequency patterns regulated by local government (i.e., 390Hz and its harmonics in U.S.). Traditionally, mufflers are designed at a large granularity, aiming to filter sound in a wide band of frequency range, partially because of its ease of control using relatively simple muffler geometries.

Engine noise muffler Here we demonstrate the possibility of controlling muffler behavior at finer granularity using our modular filter, because of its ability to construct complex muffler structures. We aim to construct mufflers that selectively attenuate sound near a set of discrete frequency values. Our first example is to attenuate a recorded engine noise, which has peaks in frequency domain at 850Hz, 1550Hz, and 2100Hz. To filter these frequency components, we uniformly sample frequen-

	Optimization			
	# DoFs	type	# targets	avg. time
Piggy	21	Z	3	9m
Octopus	76	TL	8	2h10m
Bob	258	Z	13	7h
EngineMuff	20	TL	3	15m
EarEngine	51	TL	3	11m
EarHorn	127	TL	7	1h15m
Hippo	122	Z	4	51m

Table 4.1: **Optimization Statistics** The number of DoFs is the sum of number of feasible nodes and number of connecting faces. Optmization time is averaged over all the optimized targets for each example. The number of targets is the number of peaks and valleys that we want to optimize in each example.

cies $\omega_i, i = 1..N_\omega$, which include the peak frequencies. We then define an objective function (4.7), in which the $g(\mathsf{T}(\omega_i))$ compute the transmission loss (using Equation (4.3)), and $\bar{g}_i$ is a large value at the peak frequencies and zero otherwise. The muffler structure is optimized with a combination of 8 resonators, and the quantitative results is plotted in Figure 4.11 (orange curve). We also compare the result with a muffler that has the same volume of the internal chamber but unoptimized structure (blue curve), showing that the optimized muffler indeed attenuates the unwanted frequency peaks. Please refer to the video for their audible differences.

Acoustic earmuffs Our next example of muffler design is for acoustic earmuffs. There has been a variety of acoustic earmuffs targeting at different application scenarios, such as hunting, construction work, and riding motorcycles. While some earmuffs also employ a microphone mounted in the headset to actively reduce broadband noise, many others reply on acoustic structures and materials for noise reduction and have the advantage of robustness and long working time (without any battery). Our method can also design passive earmuffs, but complement this category by allowing

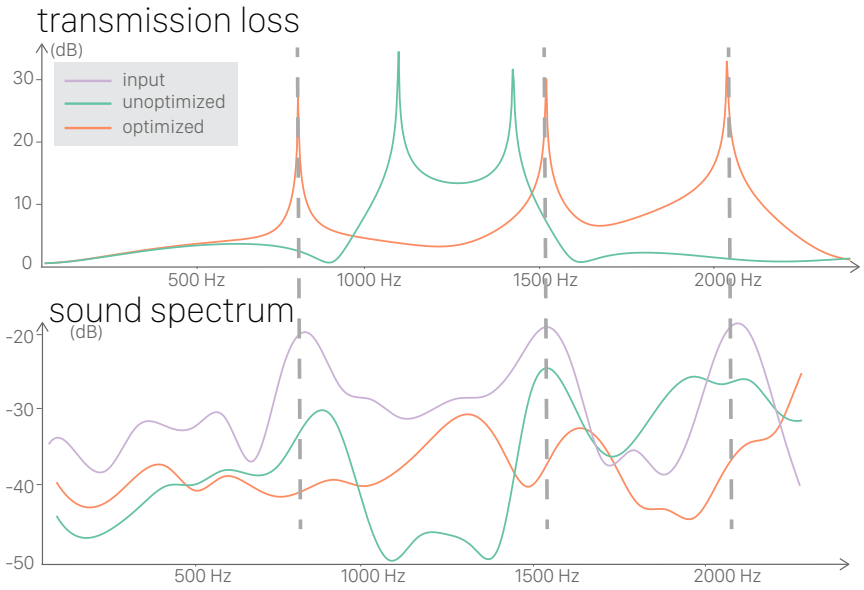


Figure 4.11: **Engine Muffler**. We compare an unoptimized muffler and an optimized one. The three noisy peaks are suppressed to lower levels with the optimized muffler.

user customization.

We demonstrate two earmuffs that can be modularly mounted in the headset (see video) and switch to different ones when needed. The first one is customized to reduce engine noise having peak frequencies at 1000Hz, 1600Hz, and 2200Hz (Figure 4.12-top). The second one is designed for riding motorcycles (Figure 4.12-bottom). We aim for reducing aerodynamic noise while allowing the rider to hear car horns for the sake of safety. Therefore, the objective function is to suppress a broadband noise without heavily filtering car horn sound at 390Hz and its harmonics. Both earmuffs are computed by optimizing the structure of 42 primitive resonators. As shown in the plots of Figure 4.12, our mufflers indeed filter out frequency components we desired.

4.5.3 Application II: Wind Instruments

Acoustic resonator is a key part of wind instruments. While nonlinear excitation mechanism of a wind instrument (such as the mouth piece) is also important [Allen and Raghuvanshi, 2015], criti-

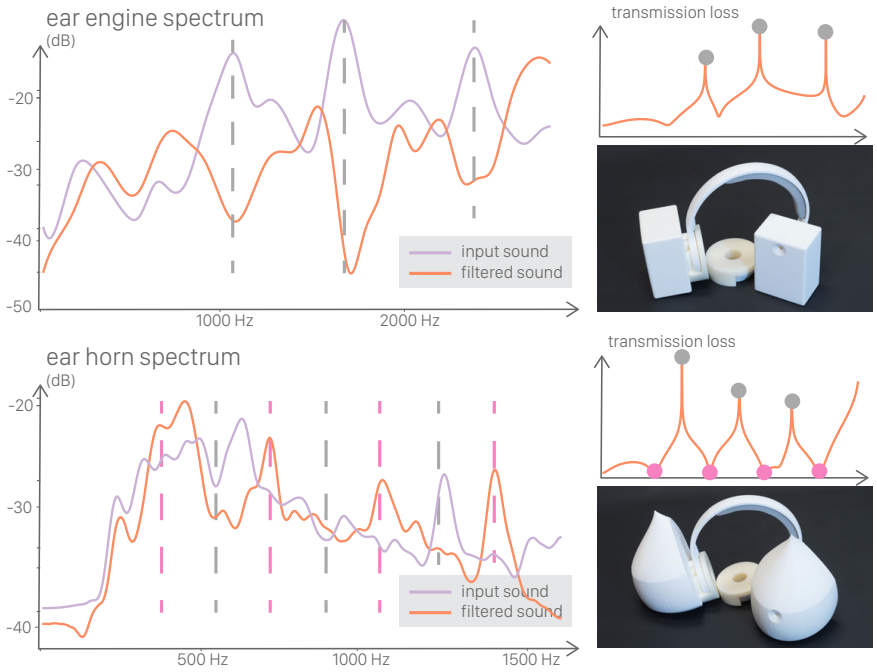


Figure 4.12: **Acoustic earmuff**. We customized two earmuffs (top and bottom) that can be modularly mounted in a headset. In the plots on the left, the orange curves show the filtered sounds where the peaks and valleys correspond to the purple points on the right.

cally affecting the timbre of the instrument, the acoustic resonator serves to modulate the excitation and controls the pitch. In particular, it is known that the playable notes of a wind instrument correspond to the peaks of the resonator's input impedance, except its first peak (called pedal note).

We applied our method to customize trumpets. Our customization is twofold: we wish to control the set of notes that a trumpet can play while customizing its shape, which, in our case, a cartoon hippopotamus shape. The resulting trumpet still relies on the standard mouthpiece for excitation. Given a set of notes, we define an objective function (4.7) that maximizes the impedance values at the frequencies of those notes. We customized 3 different trumpets, whose playable notes are [G4,D5], [C4, G4, C5], and [G4, $B^{\flat}4$, $C^{\#}5$, E5], respectively. As shown in Figure 4.13, our optimized primitive assembly can be placed inside of the hippopotamus shape and are playable. In the supplemental video, we demonstrate that the resulting musical notes produced by our customized resonators are in tone, whereas the unoptimized resonator deviates a lot from the desired notes. We note that while

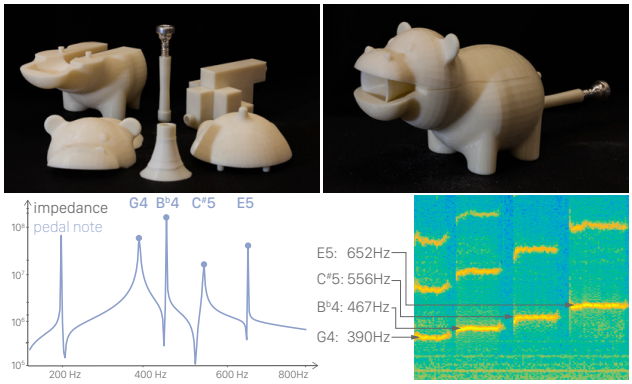


Figure 4.13: **Wind instrument**. We optimize for 4 notes for the HIPPO trumpet to play, located at the impedance maximums, the first one being the pedal note, a sustained tone. The spectrogram of our recording confirms the accuracy of our optimization framework.

it is known that the players can "bend" the notes by around a semitone, it is difficult to rely on this controllability to play in tune especially without our assembly optimization.

4.5.4 Application III: Acoustic Signatures

Our acoustic filter design opens up possibilities for new applications. Inspired by the recent work on creating tangible input devices that interact through acoustics [Laput *et al.*, 2015; Savage *et al.*, 2015], we demonstrate two examples, namely *acoustic tagging* and *acoustic encoding*.

Acoustic tagging Our method enables a new way of tagging 3D shapes. This is similar in spirit to the recent work on tagging 3D fabricated shapes by modulating material distribution and decoding using Terahertz imaging [Willis and Wilson, 2013], but from a completely different perspective, the acoustics.

Our key idea is to embed tags into the acoustic filtering effects of a shape, by computationally optimizing its internal structure without largely changing its visual appearance, as long as the shape has two holes serving as the inlet and outlet (Figure 4.1-a). Even with a single tapping using a palm at a

hole, one can produce an acoustic wave passing through the internal structure and output a filtered noise. A simple FFT-based algorithm can recognize the output sound and decode the tags. Compared to the existing tagging approaches, this method requires no electronics during installation and detection (unlike Radio Frequency Identification tags) or multi-material fabrication (unlike [Willis and Wilson, 2013]). It relies on our optimization method to physically realize a specific acoustic signature that can be reliably read by a computer program. In our examples, we choose to make each tag to have distinct peaks of their impedance curves, and thereby allowing for robust, FFT-based decoding.

We demonstrate this approach by fabricating three identical piggy shapes (Figure 4.1-b), each with an target acoustic impedance curve peaking at different frequency values (Figure 4.14). Using our Acoustic Voxels approach, we realize these impedance curves with our primitive assemblies. We have implemented a simple iPhone application that decodes a recorded tapping sound and detect the resonant frequencies which correspond to the local maximums on the impedance curve. As shown in the video (and Figure 4.1), the iPhone application can reliably detect the tags and identify the piggies.

To take the complexity of our optimized muffler further, we voxelized Bob, the duck-shaped lifesaver, with the inlet at the beak and the outlet at the tail (Figure 4.15). We optimized for two sets of frequency peaks on the impedance curve; each has more than 10 peaks. We evaluated this example by comparing the target impedance against the optimized impedance computed using our simulation model without fabricating the models, because of the 3D printer's limitation on the geometric size of the fabricated shapes (Figure 4.15). This example promises for tagging a large pool of objects or controlling the filtering behaviors at a finer granularity in future.

Acoustic encoding Taking one step further, we demonstrate the ability to encode bit strings, which can be interpreted as virtually any type of information, akin to the idea of QR code but visu-

ally less distracting. The idea is again using acoustic filter to modulate frequencies in a controlled way. Instead of controlling acoustic impedance curves, here we explore the possibility of encoding in the transmission loss curve, with a simple coding scheme: To encode N bits of information, we evenly sample 2N frequency values and group the samples pairwise. Let the frequencies are grouped as $(\omega_1, \omega_2), (\omega_3, \omega_4), ..., (\omega_{2N-1}, \omega_{2N})$. We encode a "1" at the i-th bit if the transmission loss value at ω_{2i-1} is smaller than that at ω_{2i}, and encode a "0" if the value at ω_{2i-1} is larger than that at ω_{2i} (Figure 4.16-b). By setting an objective function that maximizes and minimizes the transmission loss at corresponding frequencies, we optimize for an acoustic filter that physically realizes this coding scheme.

We fabricated three objects with an identical, octopus-like surface shape (Figure 4.16), and use them to encode different 4-bit strings, including "0000", "1001", and "0111". As shown in the video, we have implemented another iPhone application that plays a white noise from its speaker while simultaneously recording from its microphone. When aligning the iPhone speaker and microphone with two holes (i.e., the inlet and outlet) on the object, the white noise passes through the internal structure of the shape and gets filtered. By detecting the filtered amplitudes at the pre-specified frequencies ω_i, the application decodes the bit strings. In future, the application can be made to interpret the bit strings in a specific context and enable other new applications.

4.6 Conclusion

Our method is mostly suitable for controlling impedance and transmission loss at discrete frequencies, but has limited ability to control a broadband of frequencies. For this purpose, the traditional muffler design is more suitable. Currently, we use only one rigid material and optimize the filter's chamber shape, while automotive mufflers often use composite materials. It is also less clear, when optimizing for more acoustic properties, how much control can be exerted via merely the assembly

shape optimization. So far, we consider only a single type of primitive resonators. Extending our method to more primitive shapes and materials can offer a larger palette for better acoustic filtering control. Practically, we have some difficulties to ensure the internal structure of a filter being thoroughly cleaned after 3D printing, as it is hard to examine given its structural complexity.

So far, we have demonstrated the control transmission loss and impedance curves up to 4500Hz. While this is motivated by the fact that most muffler and instrument applications operate in this frequency range, we are restricted by the precomputation time needed for computing transmission matrices at higher frequencies, as it requires significantly higher finite-element resolution and longer time to solve the Helmholtz equation at higher frequencies. In the example of Bob, we optimized for more then ten peaks in the impedance curve. To control more peaks, a higher resolution of lattices is needed, leading to a much longer optimization time. It is an interesting future work to further speed up the optimization process for more detailed control of acoustic filtering.

In conclusion, we present Acoustic Voxels, a computational method that optimizes assembly of primitive resonators to realize a target acoustic filtering property, described in acoustic impedance or transmission loss. We demonstrated our algorithm with three types of applications, including muffler design, wind instruments, and a new way of customizing 3D-printed shapes with acoustic signatures. In future, this idea can be carried over to design acoustic filters at different scales, such as at high frequencies for ultrasonic imaging and at low frequencies for improving room acoustics. Further, we are interested in exploring new HCI applications enabled by acoustic signatures as well as new acoustic meta-materials enabled by computational optimization.

Algorithm 3 SMC for Resonator Assembly Optimization

procedure MODIFIED–SMC(G_N, M, threshold)
  while true **do**
    for each topology in G_N **do**
      $\hat{p}_n \leftarrow$continuous_optimization(p_n) ▷ §4.4.3
      compute the objective J_i for $\hat{p}_n$
    end for
    if best objective $>$ threshold **then**(end optimization)
    end if
    weighted sample M topologies based on the objectives
    for each topology in G_N **do** ▷ §4.4.2
      if selected **then**
        perturb connectivity of the current graph
      else
        resample a new random graph
        MCMC step: probabilistically accept the sample
      end if
    end for
  end while
end procedure

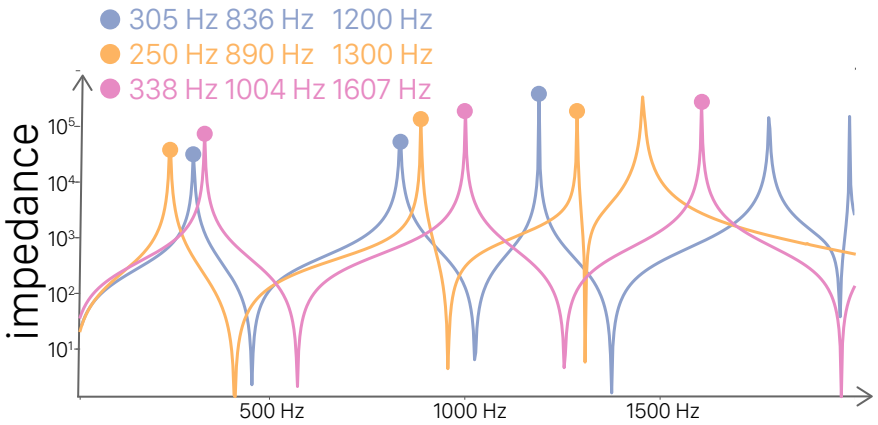


Figure 4.14: **Acoustic tagging.** We optimize three identical piggy shapes such that they all have different impedance curves. When tapped with a palm on their nose, the filtered sounds are different. The iPhone application used for recognition is shown in Fig. 4.1.

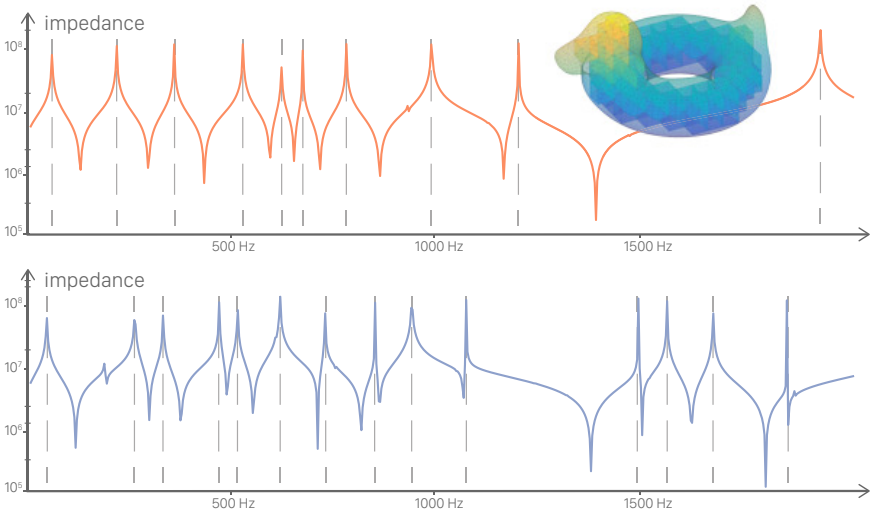
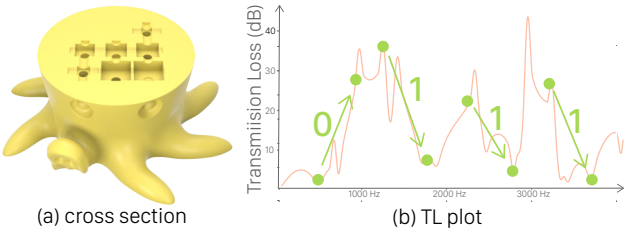
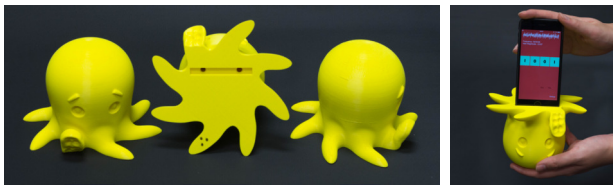


Figure 4.15: Bob. In this example we optimize for two sets of frequency peaks (top and bottom); each has more than 10 target frequency peaks, indicated by the dotted vertically lines. For both cases, out optimized acoustic filters are able to achieve the desired peaks.



(a) cross section (b) TL plot



(c) fabricated octopus with different bit pattern

Figure 4.16: **Acoustic encoding**. By embedding more voxels in the geometry, we achieve finer-grained control of the acoustic properties, exemplified by encoding 4 binary bits of information.

Chapter 5

Scene-Aware Audio for 360° Videos

5.1 Introduction

The ecosystem of 360° video is flourishing. Devices such as the Samsung Gear 360 and the Ricoh Theta have facilitated 360° video capture; software such as Adobe Premiere Pro has included features for editing 360° panoramic footage; and online platforms such as Youtube and Facebook have promoted easy sharing and viewing of 360° content. With these technological advances, video creators now have a whole new set of tools for creating immersive visual experiences. Yet, the creation of their auditory accompaniment, the immersive audio, is not as easy. Immersive 360° videos are noticeably lacking immersive scene-aware 360° audio.

Toward filling this gap, we propose a method that enables 360° video creators to easily add spatial audio from specified sound sources in a typical indoor scene, such as the conference room shown in Figure 5.1. Our method consists of two stages. We first record a single acoustic impulse response in a room using a readily available mono-channel microphone and a simple setup. Then, provided any 360° footage captured in the same environment and a piece of source audio, our method outputs the

360° video with an accompanying ambisonics spatial soundtrack. The resulting soundfield captures the spatial sound effects at the camera location, even if the camera is dynamic, as if the input audio is emitted from a user-specified sound source in the environment. Our method has no restriction on the input audio: it could be artificially synthesized, recorded in an anechoic chamber, or recorded in the same scene simply using a conventional mono-channel microphone (Figure 5.1).

A conventional microphone and a sound source are the only requirements of our method, in addition to a 360° camera. This contrasts starkly with the current approach of capturing spatial audio, which requires the use of a soundfield (ambisonic) microphone, a dedicated device that uses a microphone array (multiple carefully positioned mono microphones) to record the spatial sound field. These devices are generally expensive, and currently very few 360° cameras have an integrated ambisonic microphone. When designing sound for traditional media, audio from each source is processed to add various effects: noise removal, frequency equalization, dynamic compression, panning, and so forth. Then, the audio clips are mixed into a cohesive soundtrack for a specific layout of speakers, with a known camera angle. While ambisonics could be created virtually from these sources, it is only feasible to do this manually in a space with no reflections. In real rooms reflections off of surfaces and sources need to account for direction. Our method provides an easy way to achieve these directed reflections.

Our method enables 360° video creators to incorporate spatial audio at a lower cost, without the need of ambisonic microphones. More importantly, it allows the creator to reuse the well-established audio production pipeline, where sound effects are designed, recorded, denoised, and composed — without worrying about downstream ambisonic effects. Afterwards, our method automatically incorporates room acoustic effects in the video-shooting scene, and converts the sound produced in the earlier stage into spatial audio, which is fully synchronized with the camera trajectory in the 360° video.

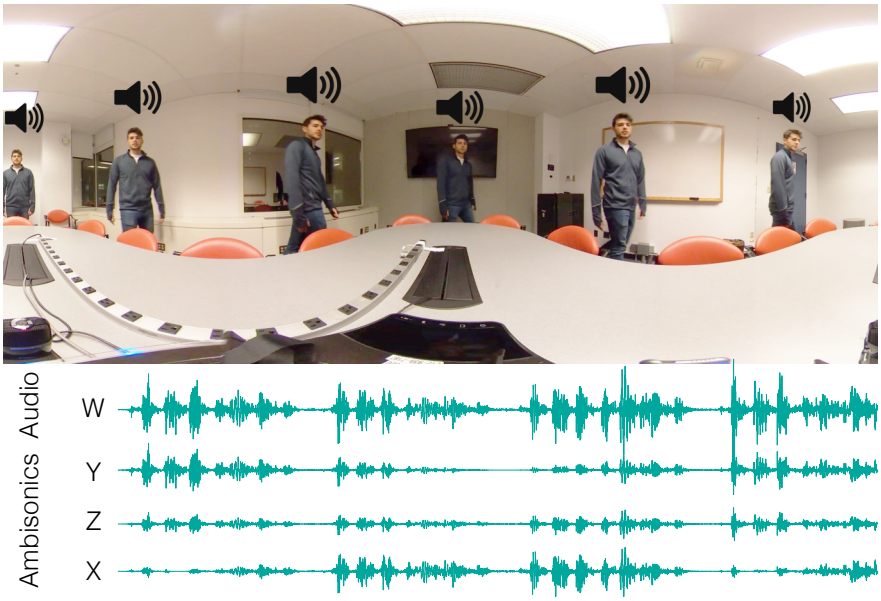


Figure 5.1: **360° audiovisual capture.** Our method enables video creators to add ambisonic audio (bottom) in a 360° video of a indoor scene (top). When viewers watch the video and change the camera angle, they hear the binaural audio consistent with the current viewing angle. Our method has no restriction on the input audio. In the case shown here, the input audio is the person's speech captured using a conventional mono-channel microphone collocated with the 360° camera, and our method converts the mono-channel input audio into a spatial audio in standard ambisonic format. The waveforms show a first-order ambisonic output (four channels), although our method supports an arbitrary order of ambisonics.

Technical insight and contributions We propose to produce spatial audio by combining a lightweight measurement of room acoustics and a fast geometric acoustic simulation. A key step in our method is to construct *directional* impulse response (IR) functions. For traditional, non-spatial audio, an acoustic IR (see Figure 5.2) is the sound recorded omni-directionally at a listening location due to an impulsive signal at a source location. Then, given any input sound signals at the source, the received non-spatial sound signals can be computed by convolving the input signals with the IR. However, to produce spatial audio, we need instead directional IR functions that record the IR sound coming from each direction at the listening location. Even in the same scene, the IR varies with respect to the source and listening locations, and the directional IR further depends on incoming sound directions.

An interesting property of IR functions lays the foundation for our proposed method. The late part of the IR is the received sound energy after excessively interacting with the scene. Every time sound waves reach a scene object, a portion of their energy is reflected "diffusely", effectively causing the sound energy distribution in the scene to become more uniform. Consequently, it is generally accepted that the late part of the IR is independent of the source and listening locations [Kuttruff, 2017]. Further, in directional IRs, the late part becomes isotropic (independent of incoming direction), as confirmed in our room acoustic measurements (see Figure 5.3 and §5.5.1).

Exploiting this property, we measure a single non-spatial IR in the scene and extract its late part through a novel method, which identifies when its energy distribution becomes truly uniform. This enables us to reuse the measured late IR when constructing the spatial IR at given source and listening locations, only relying on geometric acoustic simulation to generate the early part of the spatial IR. The simulated early IR part is further improved by a simple and effective frequency modulation method that accounts for room resonances.

To leverage acoustic simulation, we reconstruct rough scene geometry from the 360° video footage, using a state-of-the-art 360° structure-from-motion method, guided by a few user specifications. We develop an optimization approach that estimates the acoustic material properties associated with the geometry, based on the measured IR. The geometry and material parameters enable the acoustic simulation to capture the early, directional component of the spatial IR. Because the early part of the IR is oftentimes very short (typically 50-150 ms), the sound simulation is fast.

We demonstrate the quality of our resulting audio by comparing with spatial audio directly recorded by ambisonic microphones, and show that their differences are almost indistinguishable. Unlike ambisonic recordings, our method requires only a low-cost microphone, and offers the flexibility to add, replace, and edit spatial audio for 360° video. We explore the potential use of our method in several applications. While our method is designed for indoor 360° video, we further explore its use for those shot in outdoor spaces.

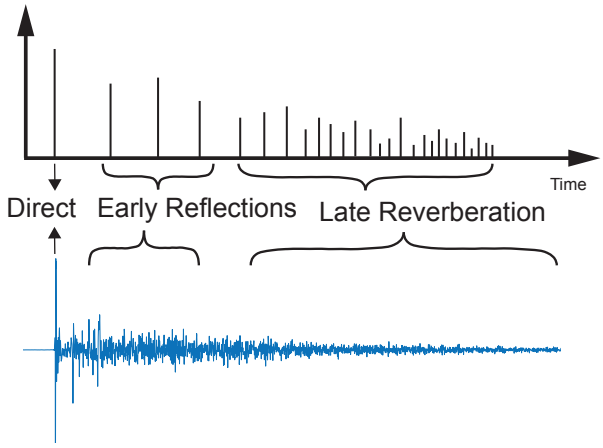


Figure 5.2: **A typical impulse response.** (top) An idealized illustration, showing the arrival time of rays and the amount of energy they carry. (bottom) A recorded impulse response in a lecture hall. The reflections become more dense and diffuse towards the later part. Traditionally, an IR is measured by recording sound omni-directionally. But for spatial audio generation, we need to estimate a *directional* IR, which is illustrated in Figure 5.9.

5.2 Background for 360° Audio

An important concept used throughout our method is the acoustic impulse response (IR). We therefore start by discussing its properties in typical indoor scenes to motivate our algorithmic choices presented later.

5.2.1 Properties of Room Acoustic Impulse Response

The room acoustic IR is a time-dependent function, describing the sound signals recorded at a listening location due to an impulsive (Dirac delta-like) signal at a source (Figure 5.2). In this thesis, we use $H(t)$ to denote an IR. If $H(t)$ is known, then the sound signal $s_{\mathrm{r}}(t)$ received at the listening location can be computed by convolving $H(t)$ with the sound signal $s_{\mathrm{e}}(t)$ emitted from the source: $s_{\mathrm{r}}(t) = s_{\mathrm{e}}(t) * H(t)$. Therefore, to add spatial audio to a 360° video, we need to estimate the IRs between the sound source and the camera location in the scene along all incoming directions.

The IR is usually split into three parts: *i)* the direct sound traveling from the source to the listener,

ii) the first few early reflections (ER), and *iii)* the later reflections called late reverberation (LR). Part (i) and (ii) are the early reflection impulse response (ERIR). Perceptually, they give us a directional sense of the sound source, known as the precedence effect [Gardner, 1968].

The LR part of the impulse response (referred to as LRIR) has several properties significant to our goal. First, the LRIR is greatly "diffused" in the scene [Kuttruff, 2017], meaning that it has little dependence with respect to the source and listening locations. This is because whenever a directional sound wave encounters an obstacle, a portion of the energy is reflected diffusely, spreading the sound in many directions. Virtually all rooms include some diffuse reflection even when the walls appear smooth [Hodgson, 1991]. Thus, the longer the sound travels in a scene, the more it gets diffused. The LRIR has little perceptual contribution to our sense of directionality. Rather, it conveys a sense of "spaciousness" [Kendall, 1995] — the size of the room, but not where the listener and source are.

Another important implication of LRIR being diffused is that the sound energy carried by LRIR tends to be uniformly distributed, not only spatially [Kuttruff, 2017] but also directionally — it can be assumed isotropic. We justify this assumption with directional acoustic measurements, as described in Figure 5.3.

5.2.2 Method Overview

The room acoustic IR properties suggests a hybrid approach for estimating spatial IRs when we generate spatial audio for 360° videos: the LRIR can be measured at one pair of source and listening locations because of its spatial and directional independence, while the ERIR needs to be simulated with carefully chosen parameters to capture sound directionality. Also crucial is the time duration for separating ER from LR, in order to ensure the directional independence of LRIR satisfied. The major steps of our method are summarized as follows.

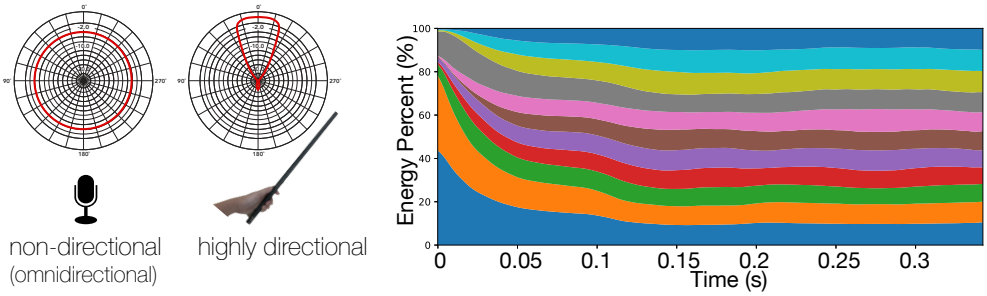


Figure 5.3: **LRIR isotropy.** We use a high-end directional (shotgun) microphone (RODE NTG8) to measure room acoustic IR received along particular directions. (left) The polar pickup plot of our shotgun microphone in comparison to the conventional omni-directional microphone. The shotgun microphone records sound mainly from its front direction. (right) For several recordings of an impulse with the shotgun microphone pointed in different directions (corresponding to different colors), we plot the amount of energy coming from each direction with respect to time. In the early part, more energy is in directions that face the source, but the energy is quickly distributed uniformly among all directions.

360° video analysis. Provided a 360° video, we estimate rough scene geometry and the camera trajectory in the scene. The former is for running the simulation, and the latter is to locate the listener when we generate spatial audio. 3D scene reconstruction has been an active research area in computer vision. We adopt the recent structure-from-motion approach [Huang *et al.*, 2017] that generates a point cloud of the scene from a 360° video (see top of the adjacent figure), along with an aligned camera path through it. Our method does not depend on this particular approach: any future improvement could be easily incorporated. Then, we rely on the user to specify a few planar shapes that align with the point cloud to capture the main geometry of the scene, such as the roof, floor, and walls (see bottom of the adjacent figure). A benefit of our hybrid approach is that only approximate geometry is required: it does not need to be water-tight, or even manifold. The method of [Huang *et al.*, 2017] takes 10-20 minutes, depending on the video resolution. Creating planar geometry to match the reconstructed point cloud only takes users several minutes per room.

Room IR analysis. Next, we record an impulse response in the room using a conventional omni-directional microphone and speaker (§5.3.1). This measurement is straightforward and serves two purposes. First, it provides the LR component when we estimate the spatial IR between a sound

source and a camera location. Second, it offers a means to sense the acoustic material properties in the room. Based on the measured IR, we estimate the acoustic material parameters for use in acoustic simulation, by formulating a nonlinear least-squares optimization problem (§5.3.2). After acquiring scene material parameters, we are then able to leverage the acoustic simulation to determine the transition point between the ERIR and LRIR based on a *directionality* analysis of the incoming sound energies (§5.3.4).

Spatial audio generation. Lastly, we generate spatial 360° audio from input audio signals. As audio editors place sources in the scene, our simulator computes the ERIR from the source to positions on the reconstructed camera path (providing directional cues), and the LRIR is reused from the measured IR (providing a sense of spaciousness). Combining them together, we obtain spatial IRs for generating spatial audio (§5.4). We show how to store the final audio in ambisonics in §5.4.2, which allows adaptation of sound effects to viewing direction when playing the final 360° video.

5.2.3 Room Acoustic Simulation

Before diving into our technical details, we briefly describe the acoustic simulator that we use. We use a geometric acoustic (GA) model that describes sound propagation using *paths* along which sound energy propagates from the source to the receiver, akin to the propagation of light rays through an environment. Each path carries a certain amount of sound energy, and arrives at the receiver with a time delay proportional to the path length. Exploiting the sound energy carried by the paths and their arrival time, we are able to infer scene materials (§5.3.2), determine ER duration (§5.3.4), and synthesize ERIRs for ambisonic audio generation (§5.4.1).

Our method does not depend on any particular GA method. In this thesis, we employ the bidirectional path tracing method recently developed in [Cao *et al.*, 2016]. This technique simulates sound propagation by tracing paths from both the sound source and the receiver, and uses multiple

importance sampling to connect the forward and backward paths. It offers a considerable speedup over prior GA algorithms and better balance between early and late acoustic responses.

While the GA model is an approximation of sound propagation and ignoring wave behaviors such as diffraction, it can reasonably estimate the impulse response of room acoustics, and has been widely used for decades [Savioja and Svensson, 2015]. Nevertheless, we consider an important wave effect, namely *room resonance*, and propose a frequency modulation method to incorporate the room resonance effect in our simulated ERIR (§5.3.3).

5.3 Room Acoustic Analysis for 360° Scenes

This section presents our method of analyzing an IR measurement to estimate acoustic material properties of the room and frequency modulation coefficients needed for compensating room resonances. We also determine the transition point between ERIR and LRIR.

5.3.1 IR Measurement

There exist many methods for acoustic IR measurement. We refer the reader to [Kuttruff, 2017] for a comprehensive summary. In this work, we use the reliable sine sweep technique of [Farina, 2000; Farina, 2007]. We briefly summarize its theoretical foundation here: the signal $s_{\mathrm{r}}(t)$ recorded by a receiver is the convolution of the source signal $s_{\mathrm{e}}(t)$ and the room's IR $H(t)$ (i.e., $s_{\mathrm{r}}(t) = s_{\mathrm{e}}(t) * H(t)$). It can be shown that $H(t)$ can be reconstructed by measuring the cross-correlation between $s_{\mathrm{r}}(t)$ and $s_{\mathrm{e}}(t)$, $H(t) = s_{\mathrm{r}}(t) \star s_{\mathrm{e}}(t)$, as long as the autocorrelation of the source signals $s_{\mathrm{e}}(t)$ is a Dirac delta, or $s_{\mathrm{e}}(t) \star s_{\mathrm{e}}(t) = \delta(t)$. For reliability, $s_{\mathrm{e}}(t)$ needs to have a flat power spectrum. A commonly used practical choice of $s_{\mathrm{e}}(t)$ is a sine sweep function that exponentially increases in

Figure 5.4: **IR measurement.** We measure the IR using a conventional speaker and a mono-channel microphone. The speaker plays a sine sweep noise, which is then recorded by a microphone. In practice, we put the speaker on soft foam to absorb any mechanical vibrations it produces, which can be propagated to the microphone through the table.

frequency from ω_1 to ω_2 in a time period T [Farina, 2000]:

$$s_{\mathrm{e}}(t) = \sin\left[\frac{\omega_1 T}{\ln\frac{\omega_2}{\omega_1}}\left(e^{\frac{t}{T}\ln\frac{\omega_2}{\omega_1}} - 1\right)\right]. \tag{5.1}$$

This signal spends more time sweeping the low-frequency regime, thus it is particularly robust to low-pass noise sources like those in most rooms. In practice, we choose $\omega_1 = 20$ Hz, $\omega_2 = 20$ kHz, and $T = 48$ seconds. Also, we play the source $s_{\mathrm{e}}(t)$ and record $s_{\mathrm{r}}(t)$ simultaneously, so they are fully synchronized. This sine sweep is played only once (no average is needed), using a conventional speaker and a mono-channel microphone. Their simple setup is illustrated in Figure 5.4.

While the IR depends on the positions of source and receiver, our measurement is insensitive to where the source and receiver are positioned. This is because for ambisonic audio generation, we only need the LR component of the measured IR, which remains largely constant in the environment (recall §5.2.1). In practice, we position the source and receiver almost arbitrarily, as long as they are well separated. We only need to perform the IR measurement once in a room. If there are multiple rooms in the 360° scene, we measure one IR per room (an example is shown in §5.5.2). This step yields a measured IR, $H(t)$, and we also compute its energy response $h(t)$.

5.3.2 Material Analysis

Having the IR measured and the room's rough geometry reconstructed from the 360° video, we now determine the acoustic material parameters needed for our room acoustic simulation. These parameters are associated with individual planar regions of the reconstructed room shape — for example, in a typical room, the walls are often painted with a particular acoustic material while the floor may have other acoustic properties. Our method also allows the user to manually select sections of the reconstructed geometry and group them as having the same acoustic material.

Acoustic properties of materials are frequency dependent. We therefore define these acoustic parameters in each octave frequency band. Without loss of generality, consider a particular octave band. When a sound wave in this octave band is reflected by a material i, part of the sound energy is absorbed by the material, which is described by the material absorption coefficient p_i in this octave band. Let $\vec{p}$ stack the p_i values of all types of materials in the room. We then formulate an optimization problem to solve for $\vec{p}$.

Path. The ray-based room acoustic simulator generates numerous paths, along which sound signals propagate from a source to a speaker. Each path is described by a sequence of 3D positions, $\vec{x}_0, \vec{x}_1, ..., \vec{x}_n$, where the first and last positions are the source and receiver, respectively. The other positions are surface points where the ray is reflected, each associated with an acoustic material (Figure 5.5). Depending on the material at position $\vec{x}_i$, $i = 1...n-1$, each $\vec{x}_i$ is mapped to an absorption coefficient indexed in the aforementioned parameter vector $\vec{p}$. Let $m(i)$ denote the index.

Energy. With this notion, the energy fraction propagated along a path j and arriving at the receiver is written as

$$e_j(\vec{p}) = \beta_j \prod_{i=1}^{N_j} \vec{p}_{m(i)}, \tag{5.2}$$

where N_j is the number of surface reflection points along the path j, and β_j accounts for the sound attenuation due to propagation in air; it depends on the path length but not on room materi-

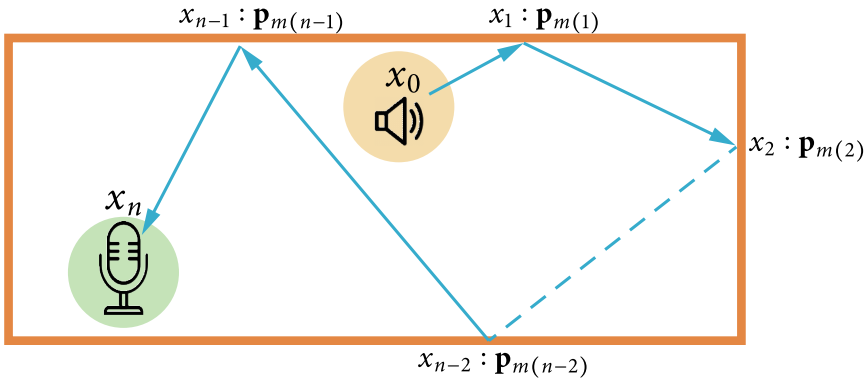


Figure 5.5: **Path and notation.** A sound path connecting a source to a receiver may be reflected multiple times at the surface positions $\boldsymbol{x}_i$. Each $\boldsymbol{x}_i$ is associated with a material indexed by $m(i)$, and its absorption coefficients over all frequency bands are stacked in the vector $\vec{p}_{m(i)}$.

als [Dunn *et al.*, 2015]. Our goal is to determine $\vec{p}$ so that the energies e_j delivered by all paths at the receiver match the energy distribution in the measured IR.

Objective function. To this end, we propose the following nonlinear least-squares objective function,

$$J(\vec{p}) = \sum_{j=1}^{M} \left[\log_{10}\left(\frac{e_j(\vec{p})}{e_0}\right) - \log_{10}\left(\frac{\tilde{h}(t_j)}{\tilde{h}(\bar{t}_0)}\right) \right]^2, \tag{5.3}$$

where $j \in [0, M]$ is the index of the paths resulted from the simulation, t_j is the sound travel time along path j, $\bar{t}_0$ is the earliest sound arrival time in the measured IR (not in the simulation), and $\tilde{h}(t_j)$ is a parametric model of the measured sound energy response at time t_j, which we will elaborate on shortly.

Moreover, e_0 is the energy delivered by the earliest path arriving at the receiver in the simulation. This is the path that directly connects the source and receiver, thus independent from material parameters. This is also the path whose arrival time is used to calibrate the reconstructed room size: before formulating the objective function (5.3), we scale the room size so that the arrival time of the first path matches $\bar{t}_0$, and in turn, the same scale is applied to the arrival time t_j of all later paths. By taking the ratio of e_j to e_0, we avoid matching the absolute energy level between the simulation

and the measurement. Later in §5.4, the energy level of simulated early rays will be scaled in order to combine with the measured LR part.

Directly using the measured energy response $h(t)$ (instead of $\tilde{h}(t)$) in (5.3) is susceptible to measurement noise. We therefore fit a parametric model $\tilde{h}(t)$ first. In this aspect, Traer and McDermott [2016] recently measured the IRs of hundreds of different daily scenes, and discovered that $h(t)$ always decays exponentially, and the decay rates are consistently frequency dependent. We therefore fit the measured $h(t)$ in each frequency band j with an exponentially decaying function, $\tilde{h}_j(t) = A_j e^{-\gamma_j t}$, and use it in (5.3), where we discard the subscript j for simplicity, as (5.3) is solved for each frequency band independently.

We note that it is critical to formulate the objective function (5.3) using a logarithmic scale, because the ray energy drops exponentially with respect to the arrival time. Otherwise, the summation in the nonlinear least-squares sense would overemphasize the match of the early paths while sacrificing late paths, which also have significant perceptual contributions [Traer and McDermott, 2016].

Inverse Solve We solve for $\vec{p}$ by minimizing (5.3) with the constraint that all values in $\vec{p}$ must be non-negative. This constrained nonlinear least-squares problem can be efficiently solved using the L-BFGS-B algorithm [Zhu *et al.*, 1997b]. This is a gradient-descent-based method, where the gradient of (5.3) is

$$\frac{\partial J}{\partial p_i} = \frac{2}{\ln 10} \sum_{j=1}^{M} \frac{1}{e_j} \left[\log_{10}\left(\frac{e_j}{e_0}\right) - \log_{10}\left(\frac{\tilde{h}(t_j)}{\tilde{h}(\bar{t}_0)}\right) \right] \frac{\partial e_j}{\partial p_i}. \tag{5.4}$$

In practice, the optimizations for individual frequency bands are performed in parallel, and often take less than 10 seconds.

As a validation, we substitute the optimized material absorption coefficient $\vec{p}$ into (5.2), and evaluate the energy e_j of every path j we collected. Using these updated e_j, we construct a simulated IR and

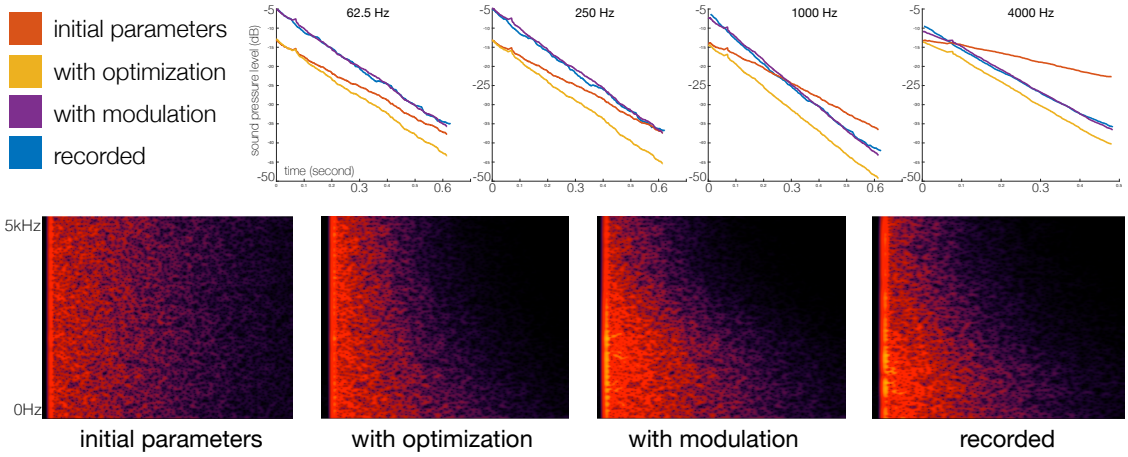


Figure 5.6: **IR optimization.** (top) The four plots correspond to four frequency bands (centered at 62.5Hz, 250Hz, 1000Hz, and 4000Hz). In each plot, the four curves correspond to the energy decay curves of four IRs obtained using different approaches. The orange curves are simulated using initial material parameters, and serve as a starting point. The blue curves are directly recorded, and are the goal. The yellow curves are simulated using our optimized material parameters. They have the same energy decay rates as the measured (blue) curves but different scales. The purple curves are computed using the yellow curves modified using our frequency modulation algorithm (see §5.3.3 and Eq. 5.7), and they match the measured curves closely. The spectrograms of the four IRs are shown on the bottom, where the simulated IR with frequency modulation matches closely with the recorded IR.

compare it with the measured IR. As shown in the top row of Figure 5.6, the energy decay rate of the *simulated* IR with respect to time indeed matches with the measured IR at every frequency band. This verifies the plausibility of our optimized parameter values. Nevertheless, the energy intensities are still different. It is this discrepancy that motivates our frequency modulation analysis, as described next.

5.3.3 Frequency Modulation Analysis

In our simulated IR, the energy decay in every frequency band always starts from e_0, the energy level delivered by the direct path from the source to the receiver. This is because the direct path has no surface reflection, and is thus independent of the material's absorption. However, this reasoning contradicts what we observe in the measured IR, where the energy decay in different frequency

bands start from different values (e.g., see the four dark blue curves in the top row plots of Figure 5.6). An important factor that cause the frequency-dependent variation is a wave behavior of sound. namely the *room resonances*. In essence, each room is an acoustic chamber. When a sound wave travels in the chamber, it boosts wave components at its resonant frequencies while suppressing others. Most rooms have their fundamental resonances in the 20-200Hz range. It is known that the room resonances affect the sound effects in the room and are one of the major obstacles for accurate sound reproduction [Cox *et al.*, 2004]. Yet, room resonance, because of its wave nature, cannot be captured by a geometric acoustic (GA) simulation.

We propose a simple and effective method to incorporate room resonances in our simulated IR. We use $\tilde{H}(t)$ to denote our simulated IR and to distinguish from the measured IR $H(t)$. Let t_0 be the arrival time of the direct path. We compute the discrete Fourier transforms of the simulated and measured IRs in a small time window Δt at t_0:

$$\tilde{\mathsf{H}}(\omega) = \mathcal{F}[\tilde{H}(t)] \text{ and } \mathsf{H}(\omega) = \mathcal{F}[H(t)], \text{ for } t_0 < t < t_0 + \Delta t. \tag{5.5}$$

Both $\tilde{\mathsf{H}}(\omega)$ and $\mathsf{H}(\omega)$ in the discrete setting are vectors of complex numbers. We compute and store the ratio $\mathsf{M}(\omega) = |\mathsf{H}(\omega)|/|\tilde{\mathsf{H}}(\omega)|$. Later, when we generate spatial IRs, we will use $\mathsf{M}(\omega)$ to modulate the frequency-domain energy of the IRs without affecting their phases (see §5.4.1).

In practice, we need to smooth $\mathsf{M}(\omega)$ in presence of measurement noise. According to the uncertainty principle of signal processing [Papoulis, 1977], we choose a small window that contains 256 samples of $\tilde{H}(t)$ and $H(t)$. This gives us 128 samples of $\mathsf{M}(\omega)$ in frequency domain. We slide the small time window Δt in a slightly larger window $[t_0, t_0 + 2\Delta t]$, repeat the computation of $\mathsf{M}(t)$, and then average the resulting ratios.

To our knowledge, methods that compensate room resonances remain elusive in existing GA-based audio generation approaches. As shown in Figure 5.6 and our supplemental video, our frequency

modulation method improves the fidelity of the simulated IR and the realism of resulting spatial audio in a very noticeable way.

5.3.4 ER Duration Analysis

After obtaining the optimized material parameters, we now use simulation to obtain a reliable estimate of the ER duration T_{ER}.

ER-LR separation is traditionally defined based on *subjective* perception [Kuttruff, 2017]. There exist various heuristics for estimating the ER duration T_{ER} from IR measurement or simulation, from a simple kurtosis threshold [Traer and McDermott, 2016] to a threshold on the number of peaks per second in a simulated IR [Raghuvanshi *et al.*, 2010a], None of these heuristics rest on the observation that we exploit to combine a simulated ERIR with a measured LRIR for ambisonic audio generation — that is, the LR is isotropic, having uniformly distributed incoming sound energy along all directions. As a consequence, simple heuristics lead to unreliable T_{ER} estimates. We therefore propose a new algorithm to determine T_{ER} directly based on the observation of the LR's isotropy.

Reusing the path energies e_j collected in §5.3.2, we define the ER duration T_{ER} as the earliest time instant when the received acoustic energy is *uniformly distributed* among all directions. To identify T_{ER}, the collected rays with their energies are viewed as Monte-Carlo samples of the energy distribution over time and direction. From this vantage point, we consider a sliding time window Δt, and check if the *statistical distance* between the energy distribution sampled by the rays in the time window and a uniform distribution is below a threshold.

Three statistical distance metrics are commonly used, including Kolmogorov-Smirnov (KS) Distance, the Earth Mover's Distance, and the Cramér-von Mises Distance. They can be viewed as taking different kinds of norms of the cumulative distribution function (CDF) difference between

two distributions. Here we choose to use KS distance, while the other two can be naturally used as well.

Our algorithm is as follows. Consider all the rays in a time window Δt=10ms. The ray directions are described by two coordinates, the azimuthal and zenith angles. We process the distribution with respect to each coordinate separately. First, we put the sampled energies e_j into histogram bins according to their zenith angles. After normalization, this histogram represents a discrete probability distribution of incoming sound energies with respect to zenith angle. We then convert this histogram into a discrete CDF, represented by a vector $\vec{P}_s$. If the energy is uniformly distributed, the expected CDF with respect to the zenith angle ϕ is

$$P_c(\phi) = \frac{1}{2}(1 - \cos\phi), \tag{5.6}$$

which is discretized into a vector $\vec{P}_c$ with the same length as $\vec{P}_s$. The KS distance is computed as $d_\phi = |\vec{P}_c - \vec{P}_s|_\infty$. Similarly, we compute the KS distance d_θ of the energy distribution with respect to the azimuthal angle θ. In this dimension, the expected CDF is simply a linear function, as θ needs to be uniformly distributed in $[0, 2\pi]$. If both KS distances are smaller than a threshold (0.15 in all our examples), we consider the current sliding time window Δt as having uniformly distributed directional energies. As we slide the time window, the first distance that passes the KS test determines T_{ER}.

To verify the robustness of this method, we run the acoustic simulation seven times, each set to produce a different number of total rays — the total number of rays increases from 15000 to 38000 evenly. After each simulation, we repeat the aforementioned analysis to compute T_{ER}. We verify that among all the T_{ER} values, the variance is small: less than 4.2% of the average T_{ER}.

5.4 Ambisonic Audio for 360° Videos

After analyzing the room geometry and acoustics, we are now able to generate ambisonic audio for any 360° video captured in the same scene. This section describes our method which produces ambisonic audio from a dry audio signal. This technique will be the cornerstone of various applications that we will explore in §5.5.2.

5.4.1 Constructing Direction-Aware Impulse Responses

Trajectory analysis Provided a 360° video, we recover the camera motion path by performing structure-from-motion analysis [Huang *et al.*, 2017]. This is the same technique that we use for reconstructing the room shape (recall §5.2.2). Our method does not critically depend on this technique; any source of geometry and a registered camera trajectory would suffice.

Simulating ER To add ambisonic sound to a 360° video, the user first clicks a location in the reconstructed 3D scene to specify a sound source position. The source location, the camera trajectory, and the room geometry together with the optimized acoustic materials provide sufficient information to launch a room acoustic simulation. The goal of this simulation is to collect a set of incoming acoustic rays at each sampled location along the camera trajectory. These rays will be used to construct directional IRs for early reverberation. Therefore, in our path-tracing acoustic simulation, we cull a path whenever its travel time exceeds T_{ER}. This restriction of simulating only early paths significantly lowers the simulation cost. In our implementation, culling paths using T_{ER} yields 10$\sim$20$\times$ speedups and memory savings in comparison to a simulation that lasts for the time length of measured IR.

In practice, we sample positions every 50 centimeters along the camera trajectory, and for each position, we collect incoming rays that arrived before T_{ER}. Each ray is described by its arrival time,

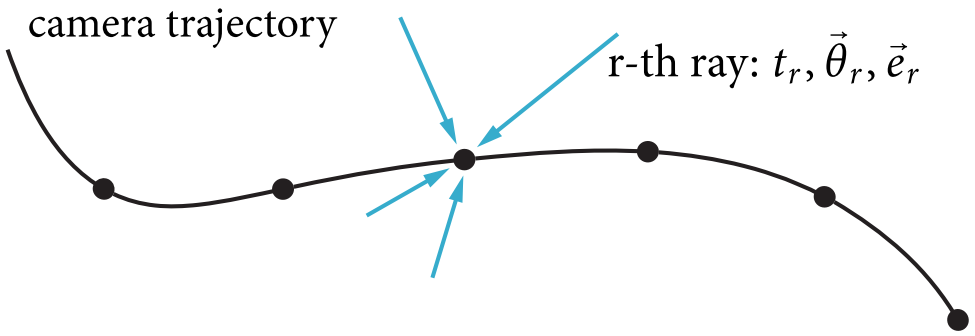


Figure 5.7: **Ray samples.** We sample locations along the camera trajectory, and use geometric acoustic simulation to collect sound rays that arrive to each location within T_{ER} after an impulsive sound signal is emitted from the source. These rays will be used for synthesizing the ERIR for spatial audio.

its incoming direction $\vec{\theta}$ (including azimuthal and zenith angles) and the carried sound energy e_i of every octave band i (see Figure 5.7).

Constructing IRs Next, at every camera position, we construct spatial IRs for ambisonic audio synthesis. Each spatial IR is decomposed into two components. The early reverberation component (ERIR) is directional, constructed individually from the simulated early rays. Given a ray r coming from the direction $\vec{\theta}$ and carrying energies $e_{r,i}$ of all octave bands (index by i), we generate an ERIR component $H^*_{r,\vec{\theta}}(t)$ using the classic Linkwitz-Riley 4th-order crossover filter, as was used in [Schissler *et al.*, 2014].

At this point, we apply the frequency modulation curve $\mathsf{M}(\omega)$ that we computed in §5.3.3 to $H^*_{r,\vec{\theta}}(t)$, because the early rays resulting from GA-based simulation do not capture the room resonances. In particular, we compute the Fourier transform of $H^*_{r,\vec{\theta}}(t)$ to get $\mathsf{H}^*_{r,\vec{\theta}}(\omega) = \mathcal{F}[H^*_{r,\vec{\theta}}(t)]$, and scale it using $\mathsf{M}(\omega)$ before transforming it back in time domain. The resulting ERIR,

$$H_{r,\vec{\theta}}(t) = \mathcal{F}^{-1}[\mathsf{H}^*_{r,\vec{\theta}}(\omega)\mathsf{M}(\omega)], \tag{5.7}$$

is what we will use for spatial audio generation (§5.4.2). As shown in the supplemental video's soundtrack, this step improves the realism of resulting spatial audio in a noticeable way.

The LRIR component $H_{\mathrm{L}}(t)$ is omni-directional, directly taken by scaling the measured IR $H(t)$ for $t > t_{\mathrm{ER}}$:

$$H_{\mathrm{L}}(t) = \begin{cases} 0, & t < t_{\mathrm{ER}} \\ \left[\left(\int_{T_{\mathrm{ER}}-\Delta t}^{T_{\mathrm{ER}}} h(t)\mathrm{d}t \right)^{-1} \sum_{r \in \mathcal{W}} \sum_i e_{r,i} \right]^{\frac{1}{2}} H(t), & t \geq t_{\mathrm{ER}}, \end{cases} \tag{5.8}$$

The scale in front of $H(t)$ is to match the energy level when combining simulated ERIR with the measured LRIR. It ensures that, in a small time window Δt near T_{ER}, the ratio of ERIR energy to LRIR energy in the synthesized IR is the same as the ratio computed using the measured energy response $h(t)$. Here, $\mathcal{W}$ denotes the set of rays whose arrival time is in the time window $[T_{\mathrm{ER}} - \Delta t, T_{\mathrm{ER}}]$, and the index i in the summation iterates through all octave bands.

5.4.2 Generating Ambisonic Audio

Lastly, provided a dry audio, we generate ambisonic audio received as the camera moves along its trajectory.

Background. Ambisonic audio uses multiple channels to reproduce the sound field arriving to a receiver from all directions. It can be understood as an approximation to the solution of the non-homogeneous Helmholtz equation,

$$(\Delta + k^2)p = -f_k(\vec{\psi}) \frac{\delta(r - r_{\mathrm{L}})}{r_{\mathrm{L}}^2}, \tag{5.9}$$

for each frequency band [Zotter *et al.*, 2009], where p is the received sound pressure, k is the wave number of the frequency band, r_{L} is the distance of sound sources from the receiver, and $f_k(\vec{\psi})$ is the directional distribution of the sound sources at the frequency band k. In our case, at each location along the camera trajectory, $f_k(\vec{\psi})$ is specified by its incoming rays. If a receiver is located at a polar

coordinate $(r, \vec{\psi})$, then its sound pressure is described by the solution of (5.9),

$$p_k(r, \vec{\psi}) = -ik \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \phi_{k,nm} Y_n^m(\vec{\psi}) h_n(kr_{\mathrm{L}}) j_n(kr), \tag{5.10}$$

where $Y_n^m(\vec{\psi})$ are the real-valued spherical harmonics, j_n are the spherical Bessel functions, h_n are the spherical Hankel functions, and $\phi_{k,nm}$ are the coefficients of $f_k(\vec{\psi})$ projected on the spherical harmonic basis,

$$\phi_{k,nm} = \int_{\mathbb{S}^2} f_k(\vec{\psi}) Y_n^m(\vec{\psi}) \mathrm{d}\vec{\psi}. \tag{5.11}$$

Equation (5.10) is the sound pressure of frequency band k. In the time domain, the received sound is a summation over all frequency bands, namely, $s(r, \vec{\psi}, t) = \sum_k p_k(r, \vec{\psi}) e^{-i\omega_k t}$, where ω_k is the frequency corresponding to the wave number k. Correspondingly, $\phi_{k,nm}$ in the frequency domain can be rewritten in the time domain using the Fourier transform,

$$\phi_{nm}(t) = \sum_k \phi_{k,nm} e^{-i\omega_k t} = \int_{\mathbb{S}^2} f(\vec{\psi}, t) Y_n^m(\vec{\psi}) \mathrm{d}\vec{\psi}. \tag{5.12}$$

where $f(\vec{\psi}, t)$ is the directional distribution of sound source signals in time domain.

In essence, ambisonic audio records the coefficients $\phi_{nm}(t)$ (normalized by a constant) up to a certain order n. At runtime, an ambisonic decoder generates audio signals output to speaker channels (such as stereo and 5.1) according to (5.10) together with a head-related transfer function model. Currently, almost all the mainstream 360° video players (such as Youtube and Facebook video players) support only first order ambisonics, which takes four channels of signals corresponding to ϕ_{nm} at $n = 0, m = 0$ and $n = 0, m = -1, 0, 1$.

Generating ambisonic channels Let $s_i(t)$ denote the dry audio signals. Using the ambisonic model, we view each early ray as a directional sound source, whose signal $s(t)$ is the dry audio

convolved with its ERIR component (i.e., $s(t) = s_i(t) * H_{r,\vec{\theta}}(t)$, where $H_{r,\vec{\theta}}(t)$ is introduced in (5.7)). Because this ray comes from direction $\vec{\theta}$, we model the corresponding $f(\vec{\psi}, t)$ in (5.12) as a Dirac delta distribution scaled by its incoming signal $s(t)$: $f(\vec{\psi}, t) = \delta(\vec{\psi} - \vec{\theta})s(t)$. Then, the audio data due to early reverberation at each ambisonic channel is

$$\phi_{nm} = \int_{\mathbb{S}^2} \delta(\vec{\psi} - \vec{\theta})s(t)Y_n^m(\vec{\psi})\mathrm{d}\vec{\psi} = Y_n^m(\vec{\theta})\left(s_i(t) * H_{r,\vec{\theta}}(t)\right). \tag{5.13}$$

In our examples, we compute ϕ_{nm} only up to the first order because of the limitation in current 360° video players. This results in four channels of signals (named as the W-, X-, Y-, and Z-channel), and their corresponding Y_n^m are $\frac{1}{\sqrt{2}}$, $\cos\theta\cos\phi$, $\sin\theta\cos\phi$, and $\sin\phi$ respectively, where θ and ϕ are the azimuthal and zenith angle of the direction $\vec{\theta}$. We iterate through all incoming rays collected in §5.4.1, compute their ϕ_{nm} using (5.13) and accumulate them into corresponding channels.

Meanwhile, the LRIR component produces audio signals $s_{\mathrm{L}}(t) = s_i(t) * H_{\mathrm{L}}(t)$. We model $s_{\mathrm{L}}(t)$ as sound signals coming uniformly from all directions according to our observation of energy isotropy in LR (recall §5.3.4). Then, $f(\vec{\psi}, t)$ in (5.12) becomes a direction independent function, $\frac{1}{4\pi}s_{\mathrm{L}}(t)$. In this case, the W-channel is accumulated by $\frac{1}{\sqrt{2}}s_{\mathrm{L}}(t)$, while the X-, Y-, and Z-channels are not affected.

After this step, the four channels of audio data are encapsulated into the 360° video. Our method can readily produce ambisonic audio with higher-order channels for future 360° video players.

5.5 Results

We present technical validation of our method, along with several useful applications. To fully appreciate our results, we encourage readers to watch our accompanying video. Our results were computed on a 4-core Intel i7 CPU. Our system, including acoustic simulation, material optimization,

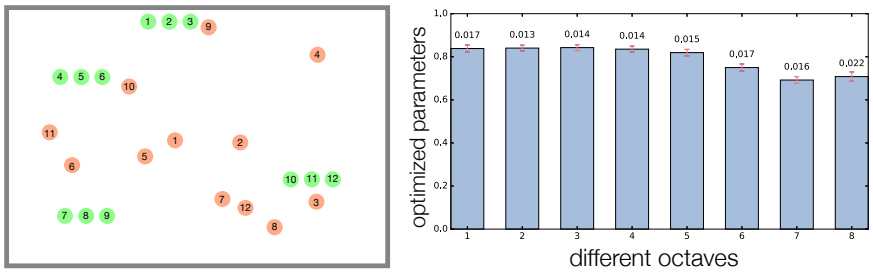


Figure 5.8: **Position independence.** We recorded 12 IRs in a room at different source and receiver locations, and perform our material estimation (§5.3.2) separately using each of the IRs. (left) We visualize the source (in green) and listening (in orange) positions used in each IR measurement indicated by the numbers inside the dots. (right) For each measurement, we optimize the material parameters, and plot the average value in each octave band (x-axis), along with error bars (indicating one standard deviation) shown on top of the bars. This plot shows that the material estimation is virtually independent from the choice of source and receiver locations.

determination of T_{ER}, frequency modulation, and ambisonic encoding, takes $\approx$ 10-20 seconds. In addition to our main supplemental video, we also provide our raw 360 videos and instructions in a supplemental zip file for full immersive experience.

5.5.1 Validation

Directionality of LRIR While the common assumption that the LRIR is diffuse spatially has been exploited in previous methods (e.g., [Raghuvanshi and Snyder, 2014]), its isotropy with respect to direction has received less attention. We therefore provide evidence through room acoustic measurement using a highly directional "shotgun" microphone. The details are described in Figure 5.3. An additional plot that also appears in the supplemental video is explained in Figure 5.9.

Robustness of material parameter estimation Part of the ease of our method rests on the fact that we only need one recorded impulse response per room using a conventional mono microphone, and that the positions of the source and receiver when recording do not matter. Figure 5.8 demonstrates the negligible impact these positions have on our IR measurement and material estimation steps. The same experiment also confirms that the LRIRs in all the measured IRs closely match each other.

This bolsters the common assumption that the LRIR is spatially diffuse.

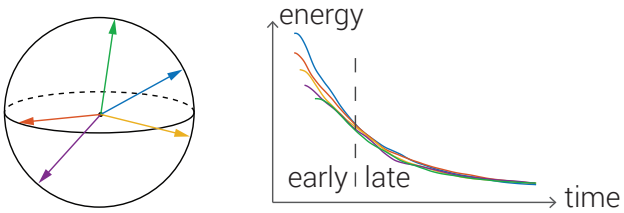


Figure 5.9: **Directional energy response.** We measure the directional energy response $h_{\vec{\theta}}(t)$ along five incoming directions (left) using a directional shotgun microphone. These measured $h_{\vec{\theta}}(t)$ (right) have different ER parts, but as time increases their LR tails converge.

Agreement with recordings We demonstrate that our algorithm can faithfully match recorded audio using ambisonic microphones. We compare recorded audio to the 360° audio synthesized by our method. In several rooms of varying size, our results match very well with the recordings (Figure 5.10). Again, please see our accompanying video to appreciate the high level of agreement our method has with recordings. To highlight the match with recordings, we stitch the recorded and synthesized audio side-by-side, to show the nearly seamless transitions.

5.5.2 Applications

Our approach enables several novel applications which make spatial audio for 360° videos easier to work with.

Audio replacement in 360° video While ambisonic microphones can be used to record spatial audio directly, they have limited use in the production pipeline. Many sounds are added to videos in post production, instead of during the video shooting. Our method allows adding sound to 360° video during post-production in a realistic spatialized fashion. We have done this in various classrooms, lecture halls, and auditoriums with varying sizes and reverberation characteristics, some of which are shown in Figure 5.10. An additional, concrete application is the removal of unwanted

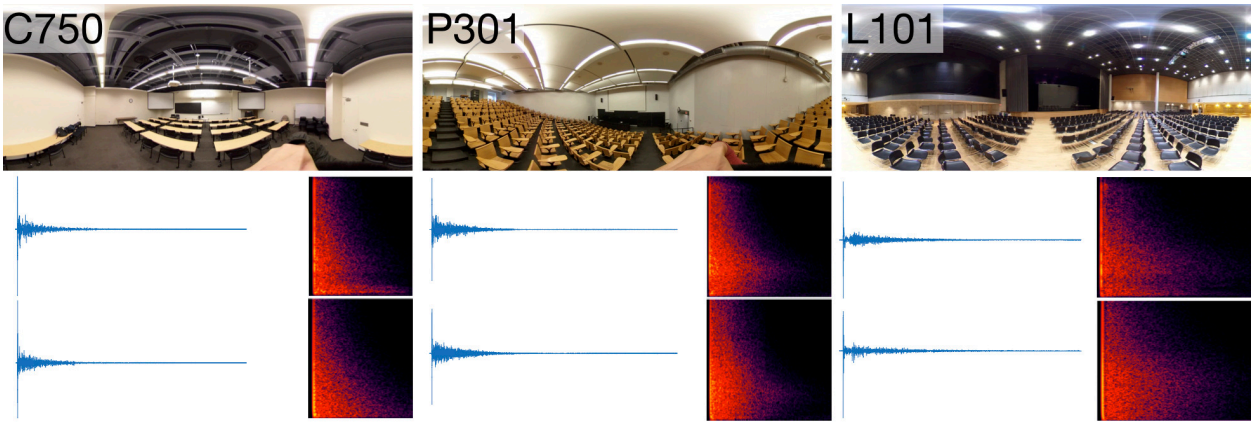


Figure 5.10: **Matching recorded IRs.** Our method (bottom) produces IRs that match closely recorded IRs (middle) for three different cases (top). Shown here are IRs of three distinct rooms. Their sizes, shapes, and materials vary largely (see Table 5.1). We refer to the supplemental materials that include audio clips of these IRs.

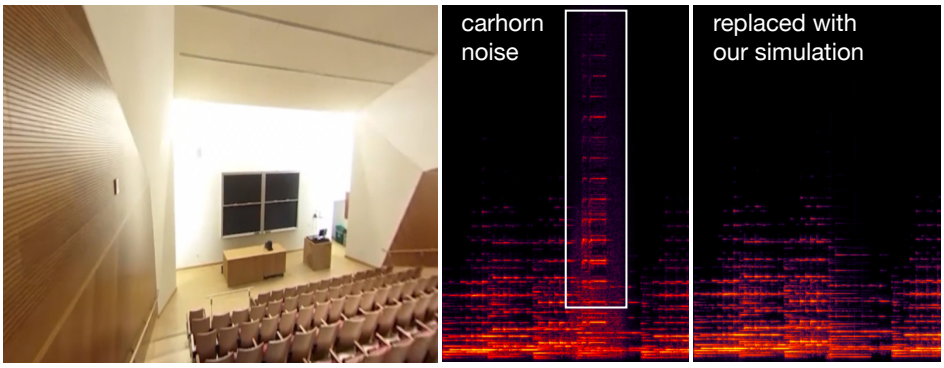


Figure 5.11: **Audio replacement.** While recording sound in a classroom, there was an unwanted car horn outside. The car horn overlapped in frequency with our desired audio, which makes removing it challenging. Using our method to resimulate the dry audio provides noise free audio that sounds as if it was recorded in the scene.

sound, shown in Figure 5.11. During one of our recordings, an unwanted car horn came from outside. Noise removal can be challenging, especially for non-stationary sources that overlap in frequency. Our method allows resimulating the desired dry audio, making it sound as if it was recorded in the same room, but with no noise.

Geometric effects One of the main benefits our method provides to 360° video editors is the ability to automatically capture geometric effects. This can easily be seen when geometry occludes the source or receiver. In this example, we moved a speaker above and below a table, causing the

	size (meter)	T_{ER}(msec)	type
C401	15×20×6	44	indoor
C620	4×6×3	21	indoor
C750	11×8×4	37	indoor
P301	12×17×7	68	indoor
N501	11×15×6	46	indoor
L101	40×60×12	121	indoor
HALLWAY	2×15×5	40	multiroom
OUTDOOR	70×50	164	outdoor

Table 5.1: **Example Statistics**

sound to become muffled. Our method captures this effect automatically (Figure 5.12). Instead of painstakingly adjusting amplitude and frequency to approximate shadowing, sound editors can now just apply a geometric filter.

Extension to cross-room propagation An even stronger geometric effect happens when a source or listener moves between rooms. This can cause very different sound due to the small opening between rooms, and different reverberation in each room. A simple extension of our method to two rooms is demonstrated (Figure 5.13). Consider a source s located in room 1 and a listening location d in room 2. Just like single rooms, the (directional) ERIR H_E^{12} between s and d is computed from simulated rays with spatial effects. For the LRIR, we recorded an IR once in each room, H^1 and H^2. We then compute the propagated IR between two rooms as

$$H_L^{12} = a \sum_{p \in A} \left(H^1_{E,\text{s}\to\text{p}} * H_L^2 + H^2_{E,\text{d}\to\text{p}} * H_L^1 + H_L^1 * H_L^2 \right), \tag{5.14}$$

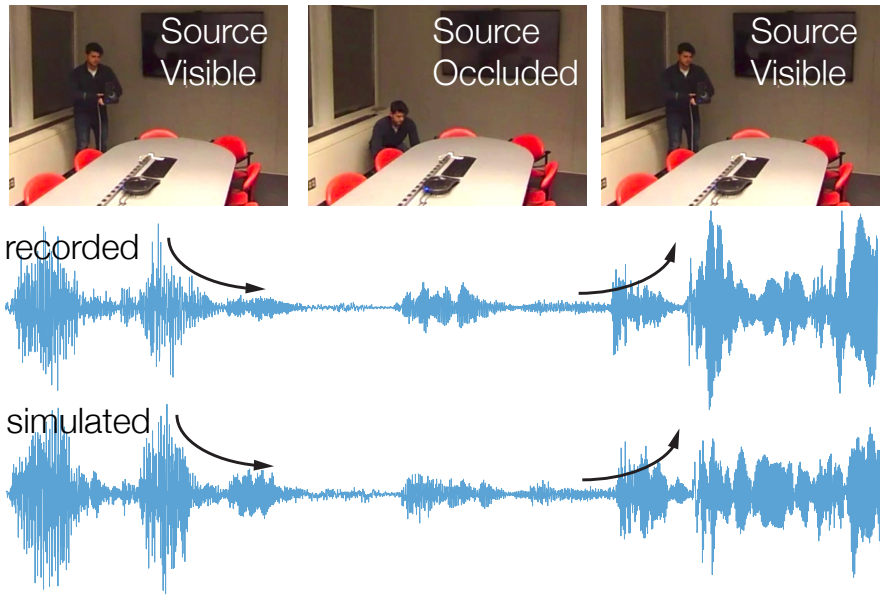


Figure 5.12: **Geometric effects: occlusion.** As a sound source moves below a table, it exhibits a low-pass muffling effect due to direct sound being blocked. Our method captures this effect.

where p are locations uniformly sampled in the planar region of the door A, a is the effective area of each sampled location p, H_L^1 and H_L^2 are the LR components of the recorded IRs in each room, and $H^1_{E,\mathrm{s}\to\mathrm{p}}$ and $H^2_{E,\mathrm{d}\to\mathrm{p}}$ are simulated ERIRs from s to p in room 1 and from d to p in room 2. The derivation of (5.14) is presented in Section B. This formulation is similar to [Stavrakis *et al.*, 2008]. Therefore, our algorithm could be easily extended to a general graph of connected rooms using their algorithm.

Re-spatialization of mono audio The final application we present is a way to apply spatial effects to *in-situ* recorded mono audio, i.e., audio recorded in a room with reverberation. This problem is similar in spirit to the conversion of a 2D film into a 3D film without refilming it — a popular problem in the film industry. Theoretically, re-spatializing the audio could be done by *deconvolving* the impulse response from the recorded audio, to obtain the original ("dry") source audio. The dry audio could then be spatialized with our method. However, deconvolution is a very ill-conditioned process and is difficult in practice. Instead, we present an ad-hoc effect that can give some spatial

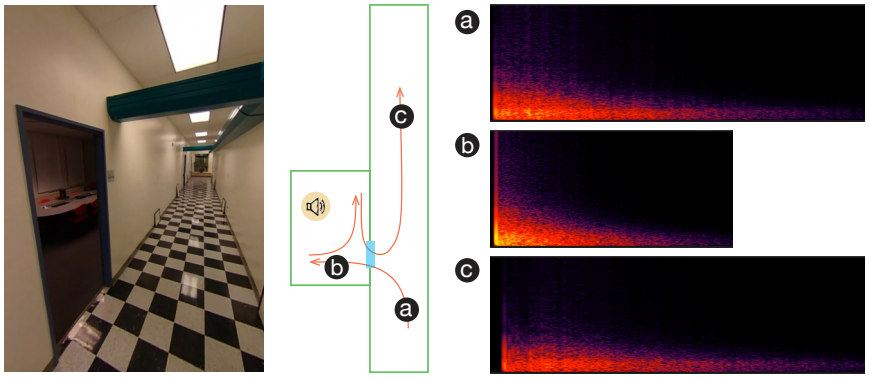


Figure 5.13: **Connected rooms.** As a listener moves between rooms, the reverberation changes, and strong geometric shadowing effects are heard. Our method naturally works in these cases, requiring only one IR measurement in each room. (a) A photograph of the multi-room scene. (b) The layout of the rooms. (c) The spectrograms of the synthesized IRs at three distinct locations.

impression. Given a room model with estimated materials, we perform a full IR simulation and store the propagated rays. We can then take the input mono-channel audio and distribute its energy over the sphere to match the energy of the computed rays. While not fully principled, it provides a plausible effect and works well in many cases, shown in Figure 5.1 and Figure 5.14.

5.6 Conclusion

We have presented a method for adding realistic, scene-aware spatial audio to 360° videos. By combining simulated early reflections with recorded late reverberation, our method is extremely fast and matches recorded audio well. It provides a practical way to incorporate geometric effects during audio post-production, requiring only a standard mono microphone and a 360° camera. We believe this will enable the next generation of sound design for emerging immersive content.

Limitations and future work A major limitation to proper viewing of spatial audio currently is the lack of personalized head-related transfer functions. These functions describe how our head and ear geometry modifies sound before it reaches our ear drums, which is how humans detect

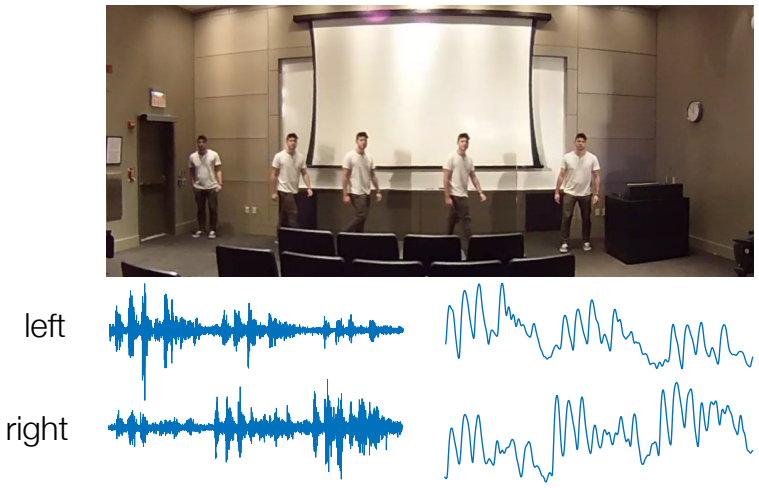


Figure 5.14: **Re-spatialization.** From recorded mono audio of a person talking while moving around a room, we can re-spatialize the sound. By using our method to compute the energy distribution due to the moving source, we distribute the mono sound energy appropriately. (top) Sound source moving from left to right in the camera frame. (bottom) The sound waveform (left) and the energy (right) after binauralizing our spatialization. Notice how the sound follows the source, moving from left to right.

directionality of sound. These functions are unique to individuals, but are laborious to measure. While the common/average models that current 360° video players use give a spatial impression, we expect the accuracy continue to increase in the future as personalized HRTFs become easier to obtain.

Our method requires a good impulse response to work well. While much easier and faster than directly measuring acoustic properties of scene materials, it is still an extra step that requires access to the original room where the video was recorded. Future work could examine inferring an impulse response from the audio in the video. Large spaces such as outdoor scenes are challenging. The large amount of uncontrollable noise makes it difficult for our method to match recordings exactly, as shown in Figure 5.15. However, this could also be seen as a strength of our method: the ability to re-simulate only the audio sources of interest, noise free.

Currently we only model the major walls and obstacles in the scene, ignoring most other objects like chairs. While it is reasonable to drop small features when the sound wave length is large enough,

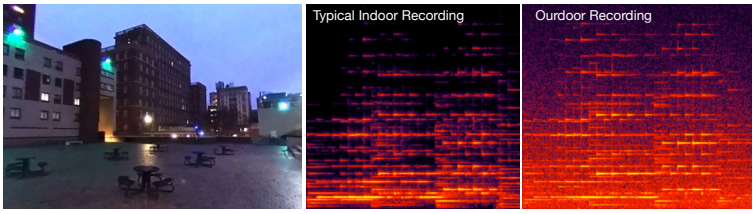


Figure 5.15: **Challenging outdoor case.** We applied our method to outdoor 360° videos. The major challenge is recording noise, due to e.g., environment and wind, which makes an exact match of our synthesized audio to the ambisonic recordings very challenging. However, the results sound plausible (see video). (left) An outdoor 360° recording scene at 6AM in the morning. (right) The recorded audio severely contaminated by noise. (middle) A typical indoor recording with much less noise.

we indeed oversimplify the reconstructed geometry. One issue occurs when the listening location becomes too close to an object that we do not model/optimize. In this case, the synthesized audio sounds may characteristically differ from the recordings. In our experiments, we found that keeping a safe distance between unmodelled objects prevents this discrepancy. In the future, we wish to investigate the impact of accurate geometric modeling on the optimization process as well as the resulting audio.

Realistic spatial audio authoring in 360° videos is an exciting and challenging research field. Thanks to recent hardware developments and surging interest in virtual reality, we expect to see an increased demand for immersive 360° audio. Our scene-aware audio is a first step towards the practical application of a more immersive audio-visual experience. In order to further advance the audio quality, still more accurate and efficient methods are required.

We believe that an intuitive spatial audio editing pipeline will go a long way to advance audio editing. Unlike mono-channel or stereo audio, high-order ambisonics have quadratically increasing number of channels, i.e., 1st order has 4, 2nd order has 9, and so on. While low-level mixing and stitching works on one or two channels for traditional audio, we argue that a higher-level abstraction of the audio editing process can help users access the full potential of spatial audio. Our work abstracts the manipulation of different channels to intuitive concepts such as the virtual sound source

location and listening location, allowing designers to think more about the scene and less about waveform editing. Lastly, we look forward to other avenues where spatial audio will enhance the user experience.

Chapter 6

Conclusion

In this thesis, I investigate the area at the intersection of virtual simulation and real-world applications. Unlike other research that has focused on solely the virtual simulation, my research considers the practical constraints and advantages imposed by hardware devices, such as 360° video cameras and 3D printers. This additional relationship with existing hardware introduced both opportunities and challenges into my sound simulation. My thesis explored different methods to address the challenges and presented a series of projects on efficient acoustic simulation.

In the following, I will briefly summarize each project and then discuss future work directions.

We have presented a physics-based interactive sound editing interface that can synthesize realistic audio for a given animation. By designing and implementing an efficient precomputation pipeline and an interactive runtime synthesis algorithm, we demonstrated the ability to change, edit, and explore new material settings on the fly. I believe this research is a cornerstone towards incorporating automatic audio generation for virtual animation.

We also introduced a new spatial audio editing pipeline that is strongly coupled together with real-world recordings, significantly improving the state-of-the-art room acoustic matching for immer-

sive audiovisual contents. We developed an inverse material estimation based visual components as well as audio recording. To further accelerate the audio composition, we observed and exploited the directional isotropy in the late tails of sound propagation. This leads to an order of magnitude speedup without compromising the audiovisual immersion.

Finally, we proposed a new acoustic filter simulation and optimization algorithm based on the idea of modularity. Modular voxels not only enable much faster forward simulations, but also make optimization possible with user-specified acoustic targets. The combination of fast simulation and inverse optimization allows one to explore the design space without a strong background in acoustics. We also manufactured all of our models and measured their acoustic filtering behavior to validate the accuracy of our efficient simulation framework.

6.1 Future Work

For discussion specific to the individual project, please refer to the end of each chapter. Here I will focus on more general discussions.

Visualization and Interfaces for Acoustic Design In order to interact with acoustic designs, one has to understand the characteristics of the current design. To convey the performance/behavior of the acoustic properties, we have adopted industry-standard metrics, such as waveform analysis, spectrum display, spectrogram visualization, and etc. However, as the goals and design domain become more complex, I think we need new effective ways to convey the performance of current designs. For example, in spatial audios, one important property is the directionality evolution in the time domain and there is no available tools or metrics to visualize it. In another fabrication scenarios, it remains With the introduction of new visualization comes the challenges on how to design better interfaces to work in tandem with the visualization methods.

Multi-Modality Fusion My thesis focuses on sound simulation. However, to reproduce our surroundings realistically in a virtual environment, it is crucial to provide multi-modality immersion beyond just auditory and visual aspects. For example, one common element missing in my research is haptic simulation. While there is a long line of research in the haptics community, it is still unexplored how to effectively simulate auditory, visual, and haptic content as well as other modalities in the same unified framework. Most existing research investigates a small subset of these modalities. As each of the components becomes more sophisticated and realistic, I believe we can learn more insights when we fuse all the modalities together.

Low-Power Mobile Simulation As mobile phones gain more computational power, there have been emerging graphics research algorithms designed with power efficiency in mind [Wang *et al.*, 2016; Wadhwa *et al.*, 2018]. More broadly, I think mobile simulation goes beyond mobile phones – smart watches, smart glasses, and other compact wearables all serve as platforms for mobile computing. While the physical sizes of these devices may shrink, the computational demands on realism and latency are higher. To some extent, we can optimize existing algorithms to adapt them to meet the power consumption threshold. However, due to the sheer power differences between mobile and desktop resources, I believe a new methodology with power consumption as part of the input constraint is an interesting research direction.

Bibliography

[Adhikari and Woodhouse, 2001] S. Adhikari and J. Woodhouse. Identification of damping: Part 1, viscous damping. *Journal of Sound and Vibration*, 243(1):43 – 61, 2001.

[Allen and Raghuvanshi, 2015] Andrew Allen and Nikunj Raghuvanshi. Aerophones in flatland: Interactive wave simulation of wind instruments. *ACM Trans. Graph.*, 34(4), July 2015.

[An *et al.*, 2012] Steven S. An, Doug L. James, and Steve Marschner. Motion-driven concatenative synthesis of cloth sounds. *ACM Transactions on Graphics (SIGGRAPH 2012)*, August 2012.

[Anderson *et al.*, 2016] Robert Anderson, David Gallup, Jonathan T Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M Seitz. Jump: virtual reality video. *ACM Transactions on Graphics (TOG)*, 35(6):198, 2016.

[Angell *et al.*, 1997] TS Angell, Xinming Jiang, and RE Kleinman. A distributed source method for inverse acoustic scattering. *Inverse Problems*, 13(2):531, 1997.

[Aubry, 2013] Jean-Pierre Aubry. *Beginning with Code_Aster*. Framasoft, 2013.

[Barbič *et al.*, 2009] Jernej Barbič, Marco da Silva, and Jovan Popović. Deformable object animation using reduced optimal control. *ACM Trans. Graph.*, 28(3):53:1–53:9, July 2009.

[Bharaj *et al.*, 2015] Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. Computational design of metallophone contact sounds.

ACM Trans. Graph., 34(6), October 2015.

[Bickel *et al.*, 2010] Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph.*, 29(4), July 2010.

[Bilbao, 2009] Stefan Bilbao. *Numerical Sound Synthesis*. John Wiley & Sons, Ltd, 2009.

[Bonneel *et al.*, 2008] Nicolas Bonneel, George Drettakis, Nicolas Tsingos, Isabelle Viaud-Delmon, and Doug James. Fast modal sounds with scalable frequency-domain synthesis. *ACM Trans. on Graph.*, 27(3), August 2008.

[Braden *et al.*, 2009] Alistair CP Braden, Michael J Newton, and D Murray Campbell. Trombone bore optimization based on input impedance targets. *The Journal of the Acoustical Society of America*, 125(4), 2009.

[Burton and Miller, 1971] A.J. Burton and G.F. Miller. The application of integral equation methods to the numerical solution of some exterior boundary-value problems. In *Proceedings of the Royal Society of London*, Series A. Math Phys Sci, pages 201–10, 1971.

[Caloz and Itoh, 2005] Christophe Caloz and Tatsuo Itoh. *Electromagnetic metamaterials: transmission line theory and microwave applications*. John Wiley & Sons, 2005.

[Cao *et al.*, 2016] Chunxiao Cao, Zhong Ren, Carl Schissler, Dinesh Manocha, and Kun Zhou. Interactive sound propagation with bidirectional path tracing. *ACM Trans. Graph.*, 35(6):180:1–180:11, November 2016.

[Chadwick and James, 2011] Jeffrey N. Chadwick and Doug L. James. Animating fire with sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4), August 2011.

[Chadwick *et al.*, 2009a] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. Harmonic shells: a practical nonlinear sound model for near-rigid thin shells. *ACM Trans. Graph.*, 28(5):119:1–

119:10, 2009.

[Chadwick *et al.*, 2009b] Jeffrey N. Chadwick, Steven S. An, and Doug L. James. Harmonic Shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Trans. on Graph.*, 28(5):1–10, 2009.

[Chiu, 2010] Min-Chie Chiu. Shape optimization of multi-chamber mufflers with plug-inlet tube on a venting process by genetic algorithms. *Applied Acoustics*, 71(6):495–505, 2010.

[Chowning, 1973] John Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, pages J. Audio Eng. Soc. 21 (7), 526–534., 1973.

[Christopoulos, 2006] Christos Christopoulos. *The Transmission-Line Modeling (TLM) Method in Electromagnetics*. Morgan & Claypool Publishers, 2006.

[Ciscowski and Brebbia, 1991] R.D. Ciscowski and C.A. Brebbia. *Boundary Element methods in acoustics*. Computational Mechanics Publications and Elsevier Applied Science, Southampton, 1991.

[Cook, 2002] Perry R. Cook. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[Corbett *et al.*, 2007] Richard Corbett, Kees van den Doel, John E. Lloyd, and Wolfgang Heidrich. Timbrefields: 3d interactive sound models for real-time audio. *Presence*, 16(6):643–654, 2007.

[Cox *et al.*, 2004] Trevor J Cox, Peter D'Antonio, and Mark R Avis. Room sizing and optimization at low frequencies. *Journal of the Audio Engineering Society*, 52(6):640–651, 2004.

[Cremer *et al.*, 2005] L. Cremer, M. Heckl, and B.A.T. Petersson. *Structure-Borne Sound: Structural Vibrations and Sound Radiation at Audio Frequencies*. Springer, 2005.

[De Lima *et al.*, 2011] Key Fonseca De Lima, Arcanjo Lenzi, and Renato Barbieri. The study of reactive silencers by shape and parametric optimization techniques. *Applied Acoustics*, 72(4):142–150, 2011.

[Dokmanić *et al.*, 2013] Ivan Dokmanić, Reza Parhizkar, Andreas Walther, Yue M Lu, and Martin Vetterli. Acoustic echoes reveal room shape. *Proceedings of the National Academy of Sciences*, 110(30), 2013.

[Dunn *et al.*, 2015] F Dunn, WM Hartmann, DM Campbell, and Neville H Fletcher. *Springer handbook of acoustics*. Springer, 2015.

[Farina, 2000] Angelo Farina. Simultaneous measurement of impulse response and distortion with a swept-sine technique. In *Audio Engineering Society Convention 108*. Audio Engineering Society, 2000.

[Farina, 2007] Angelo Farina. Advancements in impulse response measurements by sine sweeps. In *Audio Engineering Society Convention 122*, May 2007.

[Feijóo *et al.*, 2004] Gonzalo R Feijóo, Assad A Oberai, and Peter M Pinsky. An application of shape optimization in the solution of inverse acoustic scattering problems. *Inverse problems*, 20(1):199, 2004.

[Funkhouser *et al.*, 1998] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proc. of SIGGRAPH 98*, 1998.

[Funkhouser *et al.*, 1999] Thomas A. Funkhouser, Patrick Min, and Ingrid Carlbom. Real-time acoustic modeling for distributed virtual environments. In *SIGGRAPH*, pages 365–374, 1999.

[Gallivan *et al.*, 1994] K. Gallivan, E. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a lanczos method. *Appl. Math. Lett.*, 7(5):75–80, 1994.

[Gardner, 1968] Mark B. Gardner. Historical background of the haas and/or precedence effect. *The Journal of the Acoustical Society of America*, 43(6):1243–1248, 1968.

[Garland and Heckbert, 1997] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*, pages 209–216, 1997.

[Germain *et al.*, 2016] François. G. Germain, Gautham. J. Mysore, and Takako. Fujioka. Equalization matching of speech recordings in real-world environments. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 609–613, March 2016.

[Gumerov and Duraiswami, 2004] Nail A. Gumerov and Ramani Duraiswami. *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*. Elsevier Science, first edition, 2004.

[Hämäläinen *et al.*, 2014] Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. Online motion synthesis using sequential monte carlo. *ACM Trans. Graph.*, 33(4), July 2014.

[Hamming, 1983] R. W. Hamming. *Digital Filters*. Prentice-Hall, 1983.

[Hauer *et al.*, 1990] J.F. Hauer, C.J. Demeure, and L.L. Scharf. Initial results in prony analysis of power system response signals. *Power Systems, IEEE Transactions on*, 5(1):80–89, Feb 1990.

[Hodgson, 1991] Murray Hodgson. Evidence of diffuse surface reflections in rooms. *The Journal of the Acoustical Society of America*, 89(2):765–771, 1991.

[Hoppe, 1999] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *IEEE Visualization*, pages 59–66, 1999.

[Huang *et al.*, 2017] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pages 37–44, March 2017.

[Ingard, 2009] Uno Ingard. *Noise reduction analysis*. Jones & Bartlett Publishers, 2009.

[James and Pai, 2002] Doug L. James and Dinesh K. Pai. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. on Graph.*, 21(3), July 2002.

[James *et al.*, 2006a] Doug L. James, Jernej Barbic, and Dinesh K. Pai. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. on Graph.*, 25(3):987–995, July 2006.

[James *et al.*, 2006b] Doug L. James, Jernej Barbic, and Dinesh K. Pai. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph.*, 25(3), July 2006.

[Jerri, 2005] A. J. Jerri. The Shannon sampling theorem – Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, June 2005.

[Jin *et al.*, 2017] Zeyu Jin, Gautham J. Mysore, Stephen DiVerdi, Jingwan Lu, and Adam Finkelstein. VoCo: Text-based insertion and replacement in audio narration. *ACM Transactions on Graphics*, 36(4):Article 96, 13 pages, July 2017.

[Johnson and Elliott, 1995] M. E. Johnson and S. J. Elliott. Active control of sound radiation using volume velocity cancellation. *Journal of the Acoustical Society of America*, 4(98):2174–2186, May 1995.

[Kac, 1966] Mark Kac. Can one hear the shape of a drum? *American Mathematical Monthly*, pages 1–23, 1966.

[Karlsson, 1976] Johan Karlsson. Rational interpolation and best rational approximation. *Journal of Mathematical Analysis and Applications*, 53(1):38–52, 1976.

[Kaufman *et al.*, 2008] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. on Graph.*, 27(5):164:1–164:11, December 2008.

[Kausel, 2001] Wilfried Kausel. Optimization of brasswind instruments and its application in bore reconstruction. *Journal of New Music Research*, 30(1):69–82, 2001.

[Kendall, 1995] Gary S. Kendall. The decorrelation of audio signals and its impact on spatial imagery. *Computer Music Journal*, 19(4):71–87, 1995.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Klatzky *et al.*, 2000] Roberta L. Klatzky, Dinesh K. Pai, and Eric P. Krotkov. Perception of material from contact sounds. *Presence: Teleop. Virtual Environ.*, 9(4):399–410, August 2000.

[Kopf, 2016] Johannes Kopf. 360 video stabilization. *ACM Transactions on Graphics (TOG)*, 35(6):195, 2016.

[Kuttruff, 2017] Heinrich Kuttruff. *Room Acoustics*. CRC Press, sixth edition, 2017.

[Langlois *et al.*, 2014] Timothy R. Langlois, Steven S. An, Kelvin K. Jin, and Doug L. James. Eigenmode compression for modal sound models. *ACM Trans. on Graph.*, 33(4), 2014.

[Langlois *et al.*, 2016] Timothy R. Langlois, Changxi Zheng, and Doug L. James. Toward animating water with complex acoustic bubbles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016)*, 35(4), July 2016.

[Laput *et al.*, 2015] Gierad Laput, Eric Brockmeyer, Scott E Hudson, and Chris Harrison. Acoustruments: Passive, acoustically-driven, interactive controls for handheld devices. In *Proc. CHI 2015*. ACM, 2015.

[Lee *et al.*, 2016] Jungjin Lee, Bumki Kim, Kyehyun Kim, Younghui Kim, and Junyong Noh. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (TOG)*, 35(4):63, 2016.

[Lenzi *et al.*, 2013] Marcos Souza Lenzi, Sanda Lefteriu, Hadrien Beriot, and Wim Desmet. A fast frequency sweep approach using padé approximations for solving helmholtz finite element models. *Journal of Sound and Vibration*, 332:1897–1917, 2013.

[Li and Duraiswami, 2006] Zhiyun Li and Ramani Duraiswami. Headphone-based reproduction of 3d auditory scenes captured by spherical/hemispherical microphone arrays. In *IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006*, pages 337–340, 2006.

[Li *et al.*, 2015] Dingzeyu Li, Yun Fei, and Changxi Zheng. Interactive acoustic transfer approximation for modal sound. *ACM Trans. Graph.*, 35(1), December 2015.

[Li *et al.*, 2016] Dingzeyu Li, David I.W. Levin, Wojciech Matusik, and Changxi Zheng. Acoustic voxels: Computational optimization of modular acoustic filters. *ACM Trans. Graph.*, 35(4), 2016.

[Li *et al.*, 2018] Dingzeyu Li, Langlois Timothy, and Changxi Zheng. Scene-aware audio for 360° videos. *under review*, 2018.

[Lindstrom and Turk, 1998] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization*, pages 279–286, 1998.

[Liu, 2009] Y. J. Liu. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, 2009.

[Lloyd *et al.*, 2011] Brandon Lloyd, Nikunj Raghuvanshi, and Naga K. Govindaraju. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, 2011.

[Lobos *et al.*, 2003] T. Lobos, J. Rezmer, and P. Schegner. Parameter estimation of distorted signals using prony method. In *Power Tech Conference Proceedings, 2003 IEEE Bologna*, volume 4, page 5 pp, June 2003.

[Luebke, 2001] David P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35, 2001.

[Marschner and Greenberg, 1998] Stephen Robert Marschner and Donald P Greenberg. *Inverse rendering for computer graphics*. Cornell University, 1998.

[Matsumoto *et al.*, 2010] T. Matsumoto, C. Zheng, S. Harada, and T. Takahashi. Explicit evaluation of hypersingular boundary integral equation for 3-d helmholtz equation discretized with constant triangular element. *J Comput Sci Technol*, 4(3):194–206, 2010.

[Matzen *et al.*, 2017] Kevin Matzen, Michael F Cohen, Bryce Evans, Johannes Kopf, and Richard Szeliski. Low-cost 360 stereo photography and video capture. *ACM Transactions on Graphics (TOG)*, 36(4):148, 2017.

[McNamara *et al.*, 2004] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3), August 2004.

[Mehra *et al.*, 2013a] Ravish Mehra, Nikunj Raghuvanshi, Lakulish Antani, Anish Chandak, Sean Curtis, and Dinesh Manocha. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Trans. Graph.*, 32(2), April 2013.

[Mehra *et al.*, 2013b] Ravish Mehra, Nikunj Raghuvanshi, Lakulish Antani, Anish Chandak, Sean Curtis, and Dinesh Manocha. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Trans. on Graph.*, 32(2):19:1–19:13, April 2013.

[Meyer and Anderson, 2007] Mark Meyer and John Anderson. Key point subspace acceleration and soft caching. *ACM Trans. on Graph.*, 26(3), July 2007.

[Mıguez *et al.*, 2010] Joaquın Mıguez, Dan Crisan, and Petar M Djuric. Sequential monte carlo methods for the optimization of a general class of objective functions. *SIAM Journal on Optimization*, 2010.

[Monks *et al.*, 2000] Michael Monks, Byong Mok Oh, and Julie Dorsey. Audioptimization: goal-based acoustic design. *Computer Graphics and Applications, IEEE*, 20(3):76–90, 2000.

[Munjal, 2014] M.L. Munjal. *Acoustics of Ducts and Mufflers*. John Wiley & Sons, second edition, 2014.

[Noreland *et al.*, 2010] JO Daniel Noreland, M Rajitha Udawalpola, and O Martin Berggren. A hybrid scheme for bore design optimization of a brass instrument. *Journal of the Acoustical Society of America*, 128(3):1391–1400, 2010.

[O'Brien *et al.*, 2001] James F. O'Brien, Perry R. Cook, and Georg Essl. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 529–536, August 2001.

[O'Brien *et al.*, 2002] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *The ACM SIGGRAPH 2002 Symposium on Computer Animation*, pages 175–181, July 2002.

[O'Donovan *et al.*, 2007] Adam O'Donovan, Ramani Duraiswami, and Jan Neumann. Microphone arrays as generalized cameras for integrated audio visual processing. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2007.

[Pai *et al.*, 2001] Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. Scanning physical interaction behavior of 3d objects. In *SIGGRAPH*, pages 87–96, 2001.

[Panetta *et al.*, 2015] Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. Elastic textures for additive fabrication. *ACM Trans. Graph.*, 34(4), July 2015.

[Papoulis, 1977] Athanasios Papoulis. *Signal analysis*, volume 191. McGraw-Hill New York, 1977.

[Pegoraro *et al.*, 2008] Vincent Pegoraro, Ingo Wald, and Steven G Parker. Sequential monte carlo adaptation in low-anisotropy participating media. In *Computer Graphics Forum*, volume 27. Wiley Online Library, 2008.

[Pelzer and Vorländer, 2010] Sönke Pelzer and Michael Vorländer. Frequency-and time-dependent geometry for real-time auralizations. In *Proceedings of 20th International Congress on Acoustics (ICA)*, 2010.

[Pentland and Williams, 1989] Alex Pentland and John Williams. Good vibrations: model dynamics for graphics and animation. In *SIGGRAPH*, volume 23, pages 215–222, July 1989.

[Penttinen *et al.*, 2006] Henri Penttinen, Jyri Pakarinen, Vesa Välimäki, Mikael Laurson, Henbing Li, and Marc Leman. Model-based sound synthesis of the guqin. *J. of the Acoustical Society of America*, 120(6), 2006.

[Pierce and others, 1991] Allan D Pierce et al. *Acoustics: an introduction to its physical principles and applications*. Acoustical Society of America Melville, NY, 1991.

[Pillage and Rohrer, 1990] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Computer-Aided Design*, 9:352–366, April 1990.

[Pope *et al.*, 1999] Jackson Pope, David Creasey, and Alan Chalmers. Realtime room acoustics using ambisonics. In *Audio Engineering Society Conference: 16th International Conference: Spatial Sound Reproduction*, Mar 1999.

[Press *et al.*, 2007] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The art of scientific computing*. Cambridge University Press, 2007.

[Raghuvanshi and Lin, 2006] Nikunj Raghuvanshi and Ming C. Lin. Interactive sound synthesis for large scale environments. In *SI3D*, pages 101–108, 2006.

[Raghuvanshi and Snyder, 2014] Nikunj Raghuvanshi and John Snyder. Parametric wave field coding for precomputed sound propagation. *ACM Trans. Graph.*, 33(4), July 2014.

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

[Raghuvanshi *et al.*, 2010a] Nikunj Raghuvanshi, John Snyder, Ravish Mehra, Ming Lin, and Naga Govindaraju. Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes. *ACM Trans. Graph.*, 29(4), July 2010.

[Raghuvanshi *et al.*, 2010b] Nikunj Raghuvanshi, John Snyder, Ravish Mehra, Ming Lin, and Naga Govindaraju. Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes. *ACM Trans. on Graph.*, 29(4):68:1–68:11, July 2010.

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

[Ren *et al.*, 2013a] Zhimin Ren, Hengchin Yeh, and Ming C. Lin. Example-guided physically based modal sound synthesis. *ACM Trans. on Graph.*, 32(1):1:1–1:16, 2013.

[Ren *et al.*, 2013b] Zhimin Ren, Hengchin Yeh, and Ming C Lin. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG)*, 32(1):1, 2013.

[Rienstra and Hirschberg, 2003] Sjoerd W Rienstra and Avraham Hirschberg. An introduction to acoustics. *Eindhoven University of Technology*, 18:19, 2003.

[Ritchie *et al.*, 2015] Daniel Ritchie, Ben Mildenhall, Noah D Goodman, and Pat Hanrahan. Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Trans. Graph.*, 34(4), July 2015.

[Robert and Casella, 2013] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[Rubin *et al.*, 2013] Steve Rubin, Floraine Berthouzoz, Gautham J. Mysore, Wilmot Li, and Maneesh Agrawala. Content-based tools for editing audio stories. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 113–122, New York, NY, USA, 2013. ACM.

[Savage *et al.*, 2015] Valkyrie Savage, Andrew Head, Björn Hartmann, Dan B Goldman, Gautham Mysore, and Wilmot Li. Lamello: Passive acoustic sensing for tangible input components. In *Proc. CHI 2015*. ACM, 2015.

[Savioja and Svensson, 2015] Lauri Savioja and U. Peter Svensson. Overview of geometrical room acoustic modeling techniques. *The Journal of the Acoustical Society of America*, 138(2):708–730, 2015.

[Schissler *et al.*, 2014] Carl Schissler, Ravish Mehra, and Dinesh Manocha. High-order diffraction and diffuse reflections for interactive sound propagation in large environments. *ACM Trans. Graph.*, 33(4):39:1–39:12, July 2014.

[Schissler *et al.*, 2016] Carl Schissler, Aaron Nicholls, and Ravish Mehra. Efficient hrtf-based spatial audio for area and volumetric sources. *IEEE transactions on visualization and computer graphics*, 22(4):1356–1366, 2016.

[Schissler *et al.*, 2017a] Carl Schissler, Christian Loftin, and Dinesh Manocha. Acoustic classification and optimization for multi-modal rendering of real-world scenes. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[Schissler *et al.*, 2017b] Carl Schissler, Peter Stirling, and Ravish Mehra. Efficient construction of the spatial room impulse response. In *Virtual Reality (VR), 2017 IEEE*, pages 122–130. IEEE,

2017.

[Schumacher *et al.*, 2015] Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. Microstructures to control elasticity in 3d printing. *ACM Trans. Graph.*, 34(4), July 2015.

[Selamet *et al.*, 2003] A Selamet, FD Denia, and AJ Besa. Acoustic behavior of circular dual-chamber mufflers. *Journal of Sound and Vibration*, 265(5):967–985, 2003.

[Shabana, 1991] A. A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer-Verlag, New York, first edition, 1991.

[Sigmund, 1994] Ole Sigmund. Materials with prescribed constitutive parameters: an inverse homogenization problem. *International Journal of Solids and Structures*, 31(17):2313–2329, 1994.

[Siltanen *et al.*, 2008] Samuel Siltanen, Tapio Lokki, Lauri Savioja, and Claus Lynge Christensen. Geometry reduction in room acoustics modeling. *Acta Acustica united with Acustica*, 94(3):410–418, 2008.

[Smith, 1985] Julius O. Smith. A new approach to digital reverberation using closed waveguide networks. *International Computer Music Conference*, (STAN-M-31):47–53, 1985.

[Stavrakis *et al.*, 2008] Efstathios Stavrakis, Nicolas Tsingos, and Paul Calamia. Topological sound propagation with reverberation graphs. *Acta Acustica/Acustica - the Journal of the European Acoustics Association (EAA)*, 2008.

[Stettner and Greenberg, 1989] A. Stettner and D. P. Greenberg. Computer graphics visualization for acoustic simulation. In *Computer Graphics*, volume 23, July 1989.

[Strawn, 1987] John Strawn. Editing time-varying spectra. *J. Audio Eng. Soc*, 35(5):337–352, 1987.

[Takala and Hahn, 1992a] Tapio Takala and James Hahn. Sound rendering. In *Computer Graphics*, volume 26, pages 211–220, July 1992.

[Takala and Hahn, 1992b] Tapio Takala and James K. Hahn. Sound rendering. In *SIGGRAPH*, pages 211–220, 1992.

[Tan *et al.*, 2012] Jie Tan, Greg Turk, and C. Karen Liu. Soft body locomotion. *ACM Trans. on Graph.*, 31(4):26:1–26:11, July 2012.

[Tao and Seybert, 2003] Z Tao and AF Seybert. A review of current techniques for measuring muffler transmission loss. Technical report, SAE Technical Paper, 2003.

[Traer and McDermott, 2016] James Traer and Josh H. McDermott. Statistics of natural reverberation enable perceptual separation of sound and space. *Proceedings of the National Academy of Sciences*, 113(48):E7856–E7865, 2016.

[Tsingos *et al.*, 2001a] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *Proc. of SIGGRAPH 2001*, 2001.

[Tsingos *et al.*, 2001b] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 545–552, New York, NY, USA, 2001. ACM.

[Tsingos *et al.*, 2001c] Nicolas Tsingos, Thomas A. Funkhouser, Addy Ngan, and Ingrid Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH*, pages 545–552, August 2001.

[Tsingos *et al.*, 2002] Nicolas Tsingos, Ingrid Carlbom, Gary Elbo, Robert Kubli, and Thomas Funkhouser. Validation of acoustical simulations in the "Bell Labs Box". *IEEE Computer Graphics and Applications*, 22(4):28–37, June 2002.

[Tsingos *et al.*, 2007] Nicolas Tsingos, Carsten Dachsbacher, Sylvain Lefebvre, and Matteo Dellepiane. Instant sound scattering. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 111–120, 2007.

[Tsingos, 2009] Nicolas Tsingos. Precomputing geometry-based reverberation effects for games. In *Audio Engineering Society Conference: 35th International Conference: Audio for Games*, Feb 2009.

[Umetani *et al.*, 2011] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30(4):90:1–90:12, July 2011.

[van den Doel and Pai, 1996] K. van den Doel and D. K. Pai. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*, Xerox PARC, Palo Alto, November 1996.

[van den Doel *et al.*, 2001] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH*, pages 537–544, August 2001.

[Vorländer, 2008] Michael Vorländer. *Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality (RWTHedition)*. Springer, 2008 edition, 2008.

[Wadhwa *et al.*, 2018] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Trans. on Graph.*, 37(4), August 2018.

[Wang *et al.*, 2016] Rui Wang, Bowen Yu, Julio Marco, Tianlei Hu, Diego Gutierrez, and Hujun Bao. Real-time rendering on a power budget. *ACM Trans. on Graph.*, 35(4), 2016.

[Willis and Wilson, 2013] Karl D. D. Willis and Andrew D. Wilson. Infrastructs: Fabricating information inside physical objects for imaging in the terahertz region. *ACM Trans. Graph.*, 32(4), July 2013.

[Wojtan *et al.*, 2006] Chris Wojtan, Peter J. Mucha, and Greg Turk. Keyframe control of complex particle systems using the adjoint method. In *Proc. SCA*, 2006.

[Zheng and James, 2010] Changxi Zheng and Doug L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Trans. on Graph.*, 29(4), July 2010.

[Zheng and James, 2011] Changxi Zheng and Doug L. James. Toward high-quality modal contact sound. *ACM Trans. on Graph.*, 30(4), August 2011.

[Zhu *et al.*, 1997a] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4), 1997.

[Zhu *et al.*, 1997b] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997.

[Zoran, 2011] Amit Zoran. The 3d printed flute: Digital fabrication and design of musical instruments. *Journal of New Music Research*, 40(4):379–387, 2011.

[Zotkin *et al.*, 2004] Dmitry N. Zotkin, Ramani Duraiswami, and Larry S. Davis. Rendering localized spatial audio in a virtual auditory space. *IEEE Trans. Multimedia*, 6(4):553–564, 2004.

[Zotter *et al.*, 2009] Franz Zotter, Hannes Pomberger, and Matthias Frank. An alternative ambisonics formulation: Modal source strength matching and the effect of spatial aliasing. In *Audio Engineering Society Convention 126*. Audio Engineering Society, 2009.

Appendix A

Acoustic Transfer Details

A.1 Prony's Method for Transfer Computation

Consider M complex-valued transfer samples $p(\omega_t), t = 0, \ldots, M-1$ that are uniformly sampled in a frequency range $\mathcal{R}$. We seek a N-th order Prony's series to approximate it,

$$p(\omega_t) \approx \sum_{i=1}^{N} c_i e^{\mu_i \omega_t} = \sum_{i=1}^{N} c_i \lambda_i^{\omega_t}. \tag{A.1}$$

where $\lambda_i = e^{\mu_i}$. First, we define a N-th order polynomial

$$\psi(z) = \prod_{i=1}^{N} (z - \lambda_i) = z^N + a_1 z^{N-1} + \ldots + a_{N-1} z + a_N, \tag{A.2}$$

which has N roots, $\lambda_i, i = 1, \ldots, N$. And thus $\lambda_i^k \psi(\lambda_i) = 0, \forall k \geq 0$. Next, notice the equality relationship,

$$\sum_{i=1}^{N} \lambda_i^k c_i \psi(\lambda_i) = p(\omega_{N+k}) + a_1 p(\omega_{N+k-1}) + \ldots + a_N p(\omega_k) = 0. \tag{A.3}$$

This is a linear equation of $a_i, i = 1, \ldots, N$. Let k go from $[0, \ldots, M - N - 1]$. We form a linear system

$$\begin{bmatrix} p(\omega_{N-1}) & p(\omega_{N-2}) & \ldots & p(\omega_0) \\ p(\omega_N) & p(\omega_{N-1}) & \ldots & p(\omega_1) \\ \vdots & \vdots & \ddots & \vdots \\ p(\omega_{M-2}) & p(\omega_{M-3}) & \ldots & p(\omega_{M-N-1}) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} p(\omega_N) \\ p(\omega_{N+1}) \\ \vdots \\ p(\omega_{M-1}) \end{bmatrix}. \tag{A.4}$$

As long as we choose $N \leq M/2$, this is an over-constrained least-squares system with a unique solution a_i. Knowing the polynomial coefficients a_i of (A.2) allows us to find all its roots λ_i, and thus compute μ_i. Finally, after substituting λ_i into (A.1), we form another least-square system to solve c_i.

A.2 Helmholtz Boundary Element Solve

Our adaptive frequency sweeping algorithm in §3.4.2 samples frequency points and solves the Helmholtz equation. Our basic Helmholtz solver uses a BEM introduced in [Matsumoto *et al.*, 2010]. Here we sketch out the important formulas to make the thesis self-contained.

The BE solver is built upon the Kirchhoff integral formula, which is also used in [Tsingos *et al.*, 2007].

$$p(\boldsymbol{x}) = \int_S \left[G(\boldsymbol{x}; \boldsymbol{y}) \frac{\partial \phi}{\partial \boldsymbol{n}}(\boldsymbol{y}) - \frac{\partial G}{\partial \boldsymbol{n}}(\boldsymbol{x}; \boldsymbol{y}) \phi(\boldsymbol{y}) \right] dS(\boldsymbol{y}), \tag{A.5}$$

where S denote the entire object surface; $G(\boldsymbol{x}; \boldsymbol{y}) = \frac{e^{ik\|\boldsymbol{x}-\boldsymbol{y}\|}}{4\pi\|\boldsymbol{x}-\boldsymbol{y}\|}$ is the free-space Helmholtz Green's function; ϕ is the surface transfer value resulting from the BE solve (3.11); and $\frac{\partial \phi}{\partial \boldsymbol{n}}$ is the surface normal derivative of the acoustic transfer, as specified in the Helmholtz Neumann boundary condition (3.4). Once we have known the acoustic transfer $\phi(\boldsymbol{y})$ and its normal derivative $\frac{\partial \phi}{\partial \boldsymbol{n}}$ on object surface, we can use this integral formula to evaluate the transfer function at any location $\boldsymbol{x}$.

When the evaluation point is on the surface, i.e., $\boldsymbol{x} \in S$, we have the conventional boundary integral equation (CBIE),

$$\frac{1}{2}\phi(\boldsymbol{x}) = \oint\!\!\!\!\!\!-_S [G(\boldsymbol{x},\boldsymbol{y})\partial_{\boldsymbol{n}}\phi(\boldsymbol{y}) - \phi(\boldsymbol{y})\frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}]dS(\boldsymbol{y}). \tag{A.6}$$

Here we use $\oint\!\!\!\!\!\!-_S$ to indicate a Cauchy principal value at point $\boldsymbol{x}$ over the surface S. It is known that for the exterior Helmholtz problem, directly discretizing this equation using boundary elements fails to produce a unique solution at certain fictitious frequency values. Fictitious frequency results from numerical procedures and is related to the eigenfrequencies of the associated interior problem. The Burton-Miller method [Burton and Miller, 1971] takes the directional derivation of (A.6) to get a hypersingular boundary integral equation (HBIE),

$$\frac{1}{2}\partial_{\boldsymbol{n}}\phi(\boldsymbol{x}) = \oint\!\!\!\!\!\!-_S \left[\frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})}\partial_{\boldsymbol{n}}\phi(\boldsymbol{y}) - \phi(\boldsymbol{y})\frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})\partial \boldsymbol{n}(\boldsymbol{y})}\right] dS(\boldsymbol{y}), \tag{A.7}$$

and uses a linear combination of CBIE and HBIE in boundary element discretization. Formally, it solves

$$\frac{1}{2}\phi(\boldsymbol{x}) + \mathsf{D}[\phi(\boldsymbol{y})] + \beta\mathsf{H}[\phi(\boldsymbol{y})] = \mathsf{S}[\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})] + \beta\mathsf{M}[\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})] - \frac{\beta}{2}\partial_{\boldsymbol{n}}\phi(\boldsymbol{y}), \tag{A.8}$$

where the integral operators $\mathsf{D}, \mathsf{H}, \mathsf{S}$ and M are respectively

$$\mathsf{D}[\phi(\boldsymbol{y})] = \oint_S \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \tag{A.9}$$

$$\mathsf{H}[\phi(\boldsymbol{y})] = \oint_S \phi(\boldsymbol{y}) \frac{\partial^2 G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x}) \partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \tag{A.10}$$

$$\mathsf{S}[\phi(\boldsymbol{y})] = \oint_S \partial_{\boldsymbol{n}} \phi(\boldsymbol{y}) G(\boldsymbol{x}, \boldsymbol{y}) dS(\boldsymbol{y}) \tag{A.11}$$

$$\mathsf{M}[\phi(\boldsymbol{y})] = \oint_S \partial_{\boldsymbol{n}} \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})} dS(\boldsymbol{y}) \tag{A.12}$$

As long as the coefficient β has a nonzero imaginary part, this linear combination has a unique solution. A common practice is to choose $\beta = i/k$. This equation is then discretized and forms a dense linear system (3.11). In the integral equation (A.8), the surface transfer value $\phi(\boldsymbol{y})$ is unknown. Discretizing the equation (A.8) using the object's surface mesh yields a dense linear system (3.11) to solve for $\phi(\boldsymbol{y})$ on the surface.

A.3 Derivation of (3.14)

To compute the nth order expansion coefficients $\bar{p}_i$ in (3.12), we take the n-th order derivative of (3.11) at ω_0, i.e.,

$$\sum_{i=0}^{n} C_n^i \mathsf{A}^{(i)}(\omega_0) \bar{p}^{(n-i)}(\omega_0) = \boldsymbol{b}^{(n)}(\omega_0), \tag{A.13}$$

where $C_n^i = \frac{n!}{i!(n-i)!}$ are the binomial coefficients. Noticing the n-th order derivative of the expansion (3.12) is $\bar{p}^{(n)}(\omega_0) = n! \bar{p}_n$, we substitute it into (A.13) and arrive (3.14) to solve for $\bar{p}_i$.

A.4 Frequency Derivative of Linear System (3.11)

Our asymptotic waveform evaluation method involves the frequency derivative of (3.11) as derived in Section A.3. In particular, we need to compute

$$\frac{\partial^n \mathsf{A}(\omega)}{\partial \omega^n} \text{ and } \frac{\partial^n \boldsymbol{b}(\omega)}{\partial \omega^n}. \tag{A.14}$$

Their analytic forms can be computed by taking the derivatives of (A.8). In our implementation, we use piecewise constant boundary element. Therefore, $\partial_{\boldsymbol{n}}\phi(\boldsymbol{y})$ and $\phi(\boldsymbol{y})$ in (A.8) can be moved outside of the integral (i.e., the Cauchy principle value). For example, we discretize the first term of (A.8) as

$$\oint_S \phi(\boldsymbol{y}) \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \approx \sum_{\text{triangle } i} \phi(\boldsymbol{y}_i) \oint_{\Delta_i} \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}), \tag{A.15}$$

where Δ_i denotes the surface of the i-th triangle element, and $\phi(\boldsymbol{y}_i)$ is the constant sound pressure value at Δ_i. We evaluate the integral on Δ_i using a Gaussian quadrature scheme,

$$\oint_{\Delta_i} \frac{\partial G(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} dS(\boldsymbol{y}) \approx \sum_{j \text{ on } \Delta_i} w_j \frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{y}_j)}, \tag{A.16}$$

for a set of Gaussian quadrature points $\boldsymbol{y}_j$ on Δ_i. This expression also shows that to compute the frequency derivative of (3.11) analytically we need to compute

$$\frac{\partial^n}{\partial \omega^n} \left(\frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{y}_j)} \right). \tag{A.17}$$

And similarly for other integral terms in (A.8), we need to compute

$$\frac{\partial^n}{\partial \omega^n} \left(\frac{\partial G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{x})} \right), \ \frac{\partial^n}{\partial \omega^n} G(\boldsymbol{x},\boldsymbol{y}_j) \text{ and } \frac{\partial^n}{\partial \omega^n} \left(\frac{\partial^2 G(\boldsymbol{x},\boldsymbol{y}_j)}{\partial \boldsymbol{n}(\boldsymbol{x}) \partial \boldsymbol{n}(\boldsymbol{y}_j)} \right). \tag{A.18}$$

For the infinite-space Green's function $G(\boldsymbol{x}, \boldsymbol{y}) = \frac{e^{ikr}}{4\pi r}$, where $r = \|\boldsymbol{x} - \boldsymbol{y}\|_2$, the analytic normal derivatives are

$$\frac{\partial G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} = -\frac{e^{ikr}}{4\pi r^2}(1 - ikr)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{y})} \tag{A.19}$$

$$\frac{\partial G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})} = -\frac{e^{ikr}}{4\pi r^2}(1 - ikr)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{x})} \tag{A.20}$$

$$\frac{\partial^2 G(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{x})\partial \boldsymbol{n}(\boldsymbol{y})} = \frac{e^{ikr}}{4\pi r^3}[(-3 + 3ikr + k^2 r^2)\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{x})}\frac{\partial r}{\partial \boldsymbol{n}(\boldsymbol{y})} + \tag{A.21}$$

$$(1 - ikr)\boldsymbol{n}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{y})] \tag{A.22}$$

The n-th order frequency derivative of these terms are polynomials with respect to k, because $k = \omega/c$ is linear in ω and appears only in e^{ikr} and the polynomials of k in these formulas. Finally, since $\beta = i/k$ depends on ω, we compute

$$\frac{\partial^n}{\partial \omega^n}\beta = (-1)^n n! i c \omega^{-n-1}, \tag{A.23}$$

where c is the speed of sound.

A.5 Linear Solves for Mesh Simplification

The quadric error function for collapsing an edge is

$$Q^v(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{p}^T \mathsf{A}\boldsymbol{p} + \frac{1}{2}\boldsymbol{u}^T \mathsf{C}\boldsymbol{u} + \boldsymbol{p}^T \mathsf{G}\boldsymbol{u} + \boldsymbol{a}^T\boldsymbol{p} + \boldsymbol{b}^T\boldsymbol{u} + c_0, \tag{A.24}$$

where A, G, and C are 3×3 matrices, $\boldsymbol{a}$ and $\boldsymbol{b}$ are 3D constant vectors, and c_0 is a constant scalar. We refer the reader to [Hoppe, 1999] for their formulas. Initially, we minimize $Q^v(\boldsymbol{v})$ without

constraints by solving

$$\begin{bmatrix} \mathsf{A} & \mathsf{G} \\ \mathsf{G} & \mathsf{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{u} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}. \tag{A.25}$$

Next, we iteratively solve the linearly constrained quadratic programming (LCQP) problems, (3.20) and (3.21). When we solve (3.20), both $\boldsymbol{p}^T \mathsf{A} \boldsymbol{p}$ and $\boldsymbol{a}^T \boldsymbol{p}$ are constant values, and the constraint (3.19) is linear to $\boldsymbol{u}$ with a form $\boldsymbol{t}^T \boldsymbol{u} + s = 0$, where $\boldsymbol{t} = \frac{1}{6} \sum_{f \in \mathcal{N}(v)} (\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2})$ following the notations in (3.19) and $s = \frac{1}{6} \sum_{f \in \mathcal{N}(v)} \left[(\boldsymbol{p} - \boldsymbol{p}_{f1}) \times (\boldsymbol{p} - \boldsymbol{p}_{f2}) \right]^T (\boldsymbol{u}_{f1} + \boldsymbol{u}_{f2}) - C_v$. Using the method of Lagrange Multipliers, we solve this LCQP problem using a 4D linear system,

$$\begin{bmatrix} \mathsf{A} & \boldsymbol{t} \\ \boldsymbol{t}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \lambda \end{bmatrix} = - \begin{bmatrix} \boldsymbol{a} \\ s \end{bmatrix}. \tag{A.26}$$

Next, we fix $\boldsymbol{u}$ and solve the vertex position $\boldsymbol{p}$. This is an LCQP problem (3.21) with two linear equality constraints, including the volume preservation constraint $\boldsymbol{g}_{VOL}^T \boldsymbol{p} + d_{VOL} = 0$ and the volume-velocity constraint which is of a form $\boldsymbol{h}^T \boldsymbol{p} + n = 0$. Using Lagrange Multipliers, we solve a 5D linear system,

$$\begin{bmatrix} \mathsf{A} & \boldsymbol{g}_{VOL} & \boldsymbol{h} \\ \boldsymbol{g}_{VOL}^T & 0 & 0 \\ \boldsymbol{h}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = - \begin{bmatrix} \boldsymbol{a} \\ d_{VOL} \\ n \end{bmatrix}. \tag{A.27}$$

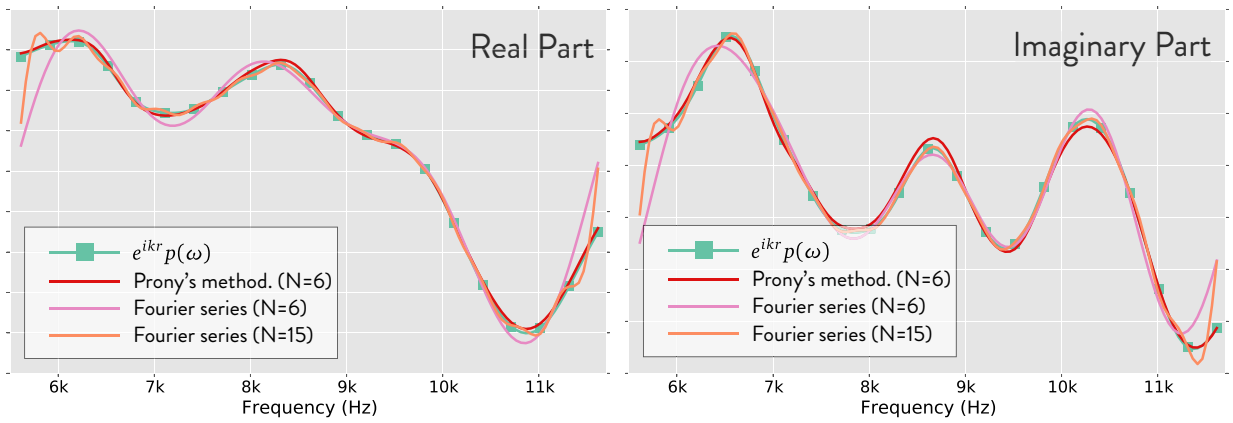


Figure A.1: **Prony Series vs Fourier Series:** We compare the approximation of frequency-dependent pressure curves using Prony series and Fourier series. The ground-truth pressure curves are the same as the ones in Figure 3.5. We plot the Prony approximation using 6 terms, and Fourier approximation using 6 and 15 Fourier basis functions respectively.

A.6 Prony Series vs Fourier Series

Here we compare the approximations using Prony series and Fourier series. As shown in Figure A.1, Prony series with just 6 terms approximate the frequency-varying pressure curves (also shown in Figure 3.5) very closely. However, the Fourier series with only 6 basis oscillate dramatically at the beginning and the ending part of the frequency window. Increasing the number of Fourier basis (even using 15 terms) still cannot completely eliminate the oscillation.

Appendix B

Cross-Room IR Formulation

Consider a source s located in room 1 and a listening location d in room 2. The IR between s and d is the result of propagating sound though the door, and thus can be written as

$$H^{12}_{\mathrm{s}\to\mathrm{d}}(t) = \int_S H^1_{\mathrm{s}\to\mathrm{p}}(t) * H^2_{\mathrm{p}\to\mathrm{d}}(t)\,\mathrm{d}S(p), \tag{B.1}$$

where S is the door area that connects two rooms (the semitransparent blue region in Figure 5.13-b), and p is a point located in the door region. $H^1_{\mathrm{s}\to\mathrm{p}}(t)$ and $H^2_{\mathrm{p}\to\mathrm{d}}(t)$ are the IRs between s and p in room 1 and between p and d in room 2, respectively. They can be approximated as concatenations of the simulated ERIR and measured LRIR in each room, namely,

$$H^1_{\mathrm{s}\to\mathrm{p}} = H^1_{E,\mathrm{s}\to\mathrm{p}} + H^1_L \text{ and } H^2_{\mathrm{p}\to\mathrm{d}} = H^2_{E,\mathrm{d}\to\mathrm{p}} + H^2_L, \tag{B.2}$$

where H^1_L and H^2_L are the LR components of the IRs recorded in each room independently (following §5.3), and $H^1_{E,\mathrm{s}\to\mathrm{p}}$ and $H^2_{E,\mathrm{d}\to\mathrm{p}}$ are simulated ERIRs between s and p in room 1 and between d and p in room 2. We note that here we use $H^2_{E,\mathrm{d}\to\mathrm{p}}$ but not $H^2_{E,\mathrm{p}\to\mathrm{d}}$ because they are the same due

to acoustic reciprocity. Then, the integrand in (B.1) becomes

$$\underbrace{H^1_{E,\mathrm{s}\to\mathrm{p}} * H^2_{E,\mathrm{d}\to\mathrm{p}}}_{\text{ERIR}} + \underbrace{H^1_{E,\mathrm{s}\to\mathrm{p}} * H^2_L + H^2_{E,\mathrm{d}\to\mathrm{p}} * H^1_L + H^1_L * H^2_L}_{\text{LRIR}}, \tag{B.3}$$

where the ERIR is replaced by with our acoustic simulation. After we discretize the door region using sampled points, the LRIR becomes the expression (5.14).

Appendix C

Supplemental Video - separately uploaded digital file