Modelling Acoustic Propagation with Dominant Paths

by

Erik Miranda

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Hearing is one of our major senses. In addition to being, arguably, our primary form of communication, it affects how we perceive the world around us. Accurate acoustic simulation can complement graphical simulation systems, providing users with more immersive and believable environments. Furthermore, sound can assist users in localizing objects within space, and it has been shown to improve the perceived quality of visuals. Many of the audio properties that our brain subconsciously keys upon are artifacts inscribed into the signal by the environment as the sound propagates from the source to the listener. Diffraction, or the bending of sound around objects, is an important mode of transmission and a key source of reverberation in an audio signal. However, many current acoustic propagation simulation systems do not properly model diffraction.

This work proposes a new method for performing the simulation of diffraction. Our formulation aims to provide accurate results across the frequency spectrum. Accordingly, we avoid using approximative models of diffraction, such as the Unified Theory of Diffraction, which are ubiquitous in the field, but lead to inaccurate results particularly in the lower frequencies. Furthermore, our algorithm can provide a time-domain solution for reverberant sounds and is controllable for stylistic adjustments. In order to evaluate our method, we compare our results against measured and experimental results reported in the literature.

Acknowledgements

A thesis is rarely an endeavour of only one person. Conversely, the number of individuals who have contributed to the process of completing this work, either directly or indirectly, is beyond my ability to accurately recall. Nevertheless, those contributions, both large and small, were crutial to the final result that you see here, and I thank you all wholeheartedly.

I think of particular merit, I must single out my supervisor, Gladimir Baranoski, who has stuck with me from beginning to end. Gladimir has been there through the various iterations of my work, from the point I had set my sights too high and gave myself a vision far beyond the scope of a Master's. He worked with me through the days of inspiration and desperation, until eventually the project culminated into a product that was fesible and finally tangible: this document that you hold now. I am sure the entire process has given him more than a couple new grey hairs, but in all senses of the phrase, I couldn't have done it without him.

I would also like to acknowledge NSERC for partially funding the work towards this thesis. Without government programs such as these, many aspiring researchers such as myself would not have the opportunities that we do today.

Dedication

This thesis is dedicated to my parents, Ana and Luiz de Miranda. For, before even the first word of the first chapter of this work was ever written, it was their dedication, love and support that brought me to that first step.

Table of Contents

List of Tables x

List of Figures xi

List of Acronyms xv

List of Symbols xvi

1 Introduction 1

2 Background 4

2.1 Principles of Sound 4

2.1.1 Speed of Sound 5

2.1.2 Amplitude of Sound 6

2.1.3 Diffraction 8

2.1.4 Impulse Response 8

2.1.5 Frequency Response 9

2.2 Diffractive Modelling 9

2.3 Ray Tracing 11

2.3.1 Visual Ray Tracing 11

2.3.2 Acoustic Ray Tracing 12

3 State of the Art **14**
3.1 Sound Synthesis . . . 14
3.2 Sound Propagation . . . 15
3.2.1 Numerical Models . . . 15
3.2.2 Geometric Models . . . 15
3.2.3 Non-Physical Models . . . 16
3.3 Diffractive Modelling . . . 16
3.3.1 Universal Theory of Diffraction . . . 17
3.3.2 Biot-Tolstoy-Medwin . . . 17
3.4 Real-Time Computation . . . 18

4 The Biot-Tolsoy-Medwin Formulation **19**
4.1 The Original Model . . . 19
4.2 Subsequent Enhancements . . . 20
4.2.1 Finite Wedge Diffraction . . . 23
4.2.2 Double Diffraction . . . 25
4.3 The BTM Line Integral . . . 25
4.4 Practical Considerations . . . 27
4.4.1 Sample-Aligned Segments . . . 29
4.4.2 Even-Sized Segments . . . 30
4.5 Our Contributions . . . 31
4.5.1 Monte-Carlo Sampling . . . 31
4.5.2 Higher-Order Diffraction . . . 34

5 Path Determination **36**
5.1 Dominant Paths . . . 36
5.2 Shortest Path Search . . . 38
5.3 Diffraction Cost Function . . . 40

5.3.1 GTD-Driven Signal Strength . . . 40
5.3.2 Distance . . . 41
5.4 Numerical Calculation of Impulse Response . . . 41

6 Implementation **42**
6.1 Preprocessing Stage . . . 42
6.1.1 Producing the Edge Cloud . . . 43
6.1.2 Producing the Edge Visibility Graph . . . 44
6.2 Path search stage . . . 44
6.2.1 Search Termination Condition . . . 45
6.2.2 Path Rejection Algorithm . . . 46
6.3 Computation stage . . . 46

7 Results **48**
7.1 Single Diffraction . . . 48
7.2 Double Diffraction . . . 50
7.3 Maekawa Finite Screen . . . 52
7.4 Computation Time and Error . . . 56
7.5 Multi-threaded Performance . . . 59

8 Conclusions **62**
8.1 Contributions . . . 62
8.2 Limitations and Future Work . . . 63

APPENDICES **64**

A Documentation **65**
A.1 Modelling API . . . 66
A.1.1 Class Vector3 . . . 66

A.1.2 Class SceneNode 66
A.1.3 Class Scene 67
A.1.4 Global Functions 67
A.2 Acoustic Simulation API 68
A.2.1 Class Edge 68
A.2.2 Class Simulator 68
A.2.3 Class Path 69
A.2.4 Class IrFunc 70
A.2.5 Class FrFunc 71
A.2.6 Class Environment 71
A.2.7 Global Functions 72

References **73**

Index **79**

List of Tables

7.1 A comparison of the accuracy of the double diffraction experiment by varying the number of samples from the original 50 million down to 0.5 million. The relative error between the 50 million sample experiment and the 5 and 0.5 million sample experiments are shown as MSE measurements in different frequency bands starting with 0 – 1 kHz and ending with 20 – 50 kHz. . . 59

7.2 A comparison of the computational time required to run the simulation of the double diffraction experiment at differing number of samples on different hardware configurations. Note that the HT column depicts the availability or setting of hyper-threading technology on the machine. 60

7.3 Comparison of the relative processing performance rates in computing the double diffraction experiment. The numbers are derived from running the 50M sample experiment. The results show significant performance gains in newer processor architectures and with the use of hyper-threading technology, as well as a linear increase in performance with increasing number of processing cores. The main features of each system are depicted in Table 7.2. 60

List of Figures

2.1 Depiction of the three phases of the IR of a finite edge diffraction event: (i) indicates the initial phase where the signal is diffracted by both upper and lower paths, (ii) indicates the half-strength phase where the signal is diffracted by only upper or only lower paths, and (iii) indicates the final phase where both the upper and lower ends are reached and no further contributions occur. 10

4.1 A view of a wedge from above. The vertical line indicates the edge of the wedge. Three paths are depicted from the source (S) to the receiver (R): the shortest path (the solid line through the apex), the longest lower path (the dashed line through z_{min}), and the longest upper path (the dash-dotted line through z_{max}). The parameters l_S and l_R describe the distances from the source and receiver respectively to the point of diffraction. The parameters r_S and z_S describe the radius from the edge and the position along the edge of the source, respectively, while the parameters r_R and z_R correspond to the radius from the edge and the position along the edge of the receiver, respectively. Finally, the parameter ϕ describes the angle which the shortest path makes with respect to the direction of the edge. 21

4.2 A cross section of a wedge from the side. A path is depicted from the source (S) to the receiver (R). The parameter θ_w represents the angle of the wedge, while the parameters θ_S and θ_R represent the wedge-to-source and wedge-to-receiver angles, respectively. 22

4.3 The phases of interaction between a wedge and the spherical wavefront emanating from the source (S) towards the receiver position (R). The wave travels toward the edge (Phase 0) and eventually intersects the edge at one point (Phase 1), causing one initial diffraction path to the receiver. As the wavefront grows, it intersects the edge twice (Phase 2), causing *upper* and *lower* diffraction paths. Finally, the wavefront will outgrow a finite edge terminating the intersections (Phase 3). In the case of an offset edge, there may not be an initial head on intersection (Phase 1b). Alternatively, there may be a period where the wavefront has outgrown the edge on one end, but not on the other (Phase 2b), leading to only an upper or a lower diffraction path. 24

4.4 The method of simulating double diffraction detailed by Medwin et al. [25]. Several points are selected on the first edge (depicted as A and B) and used as secondary sources in a single diffraction computation with the receiver (R). The aggregate result is then combined with the attenuation of the path from the source (S) to the secondary source. 26

4.5 Visual depiction of sample-aligned segments (left) versus even-sized segments (right). The upper part of the figure depicts the subdivision of the edge, where z indicates the edge position, while the lower part depicts the temporal alignment of the sampled edge-segments with the discretization of the signal function, where t indicates time. 28

4.6 A zoomed section of a single-wedge diffraction simulation to compare the error profile of different simulation algorithms. Sample-aligned segment based simulations (solid, red) provide the most accurate result for single diffraction, and serve as a reference in this example. Even-sized segment simulations (dashed, blue) produce a patterned, structural error known as *aliasing*. Finally, Monte-Carlo simulations (dotted, green) produce random error. . . 32

5.1 In 2D (left), a bend in the shortest path (indicated by the triangle) always corresponds to a vertex in the scene (indicated by the circle). In 3D (right), a bend may occur at an arbitrary point on a wedge. 38

6.1 A diagram illustrating the steps by which the preprocessing stage converts a 3D scene into a search graph to be used by the path search stage for dominant path determination. Note that the source and receiver are represented by S and R, respectively. Initially, edges on the scene (1) are subdivided using the algorithm proposed by Papadimitriou [34] (2). The visibility of each edge node to every other node is then determined (3). Finally, the resulting data is used to construct the edge visibility graph (4). 43

7.1 90° Jonasson wedge used for impulse response simulations. The source (S) and receiver (R) are both placed on opposite sides of the wedge with a 25 cm horizontal displacement and 20 cm vertical displacement from the apex. 49

7.2 Simulation of the impulse response (top) and frequency response (bottom) for an infinite (red, solid) and finite (blue, dashed) Jonasson 90° wedge. The finite wedge mid-function partial loss of strength and terminal loss of strength are clearly visible at around 2.2 ms and 2.7 ms respectively. The infinite wedge simulation shows good agreement with a pure BTM implementation reported by Medwin et al. [24] (green, dotted). 49

7.3 Experimental set-up of the double diffraction on a fixed wall scenario used by Medwin et al. [25]. The source (S) and receivers (R_{225}, R_{315}, and R_{360}) are placed 25 cm away from the nearest apex. The source and three receivers are placed at 15°, 225°, 315°, and 360° from the surface nearest to the source, respectively. The experiment is repeated for the single diffraction case for an infinitely thin wall (left), and for the double diffraction case with a wall 0.48 cm thick (right). Note that in the double diffraction 225° case, both a single (red, dashed) and a double (red, solid) diffraction path exist. 50

7.4 Simulation of double diffraction on a fixed wall as described by Medwin et al. [25]. Results for similar receiver positions are grouped together with results for the 225° case appearing at the top, the 315° case in the centre, and the 360° case at the bottom. Our results are depicted with red, green, and blue for the three cases respectively in both single (dashed) and double (solid) diffraction experiments. Results by Medwin et al. are depicted with black in both single (dotted) and double (dash-dotted) experiments. The circle, triangle, and diamond points depict measured data for the double diffraction case at the three receiver positions, respectively, as provided by Medwin et al. [25]. 51

7.5 Layout of the Maekawa finite screen experiment described by Medwin et al. [24] as seen from above (top) and from the side (bottom). Source (S) and two Receiver (R_1 and R_2) positions are depicted. 53

7.6 Orthographic rendering of the computed dominant paths (red, green, blue) for Maekawa finite screen scenario with the receiver in the first position. Simulation of reflection was achieved using the image source method [1]. Accordingly, four simulations were performed: source (S) to receiver (R), mirrored source (S') to receiver (R), source (S) to mirrored receiver (R'), and mirrored source (S') to mirrored receiver (R'). 54

7.7 Orthographic rendering of the computed dominant paths (red, green, blue) for Maekawa finite screen scenario with the receiver in the second position. Simulation of reflection was achieved using the image source method [1]. Accordingly, four simulations were performed: source (S) to receiver (R), mirrored source (S') to receiver (R), source (S) to mirrored receiver (R'), and mirrored source (S') to mirrored receiver (R'). 55

7.8 Comparison of our simulation results (red, solid) plotted against results provided by Medwin et al. [24] (black, dotted) for a Maekawa finite screen simulating two receiver positions. Both the impulse response results (top row) and frequency response results (bottom row) are depicted. In position 1 (left column), the receiver is placed under the screen, but near the edge. In position 2 (right column), the receiver is placed under the screen in a centered position. 56

7.9 Demonstration of the contributions of each dominant path to the resultant impulse response. Position 1 is depicted on the left column, and Position 2 is depicted on the right column. Starting from the top we show the source to receiver paths, the mirrored source to receiver paths, the source to mirrored receiver paths, and finally the mirrored source to mirror receiver paths. The colours of the contributions were chosen to match the rendered dominant paths in Figures 7.6 and 7.7. 57

7.10 Comparison of the accuracy of the double-diffraction experiment by varying the number of samples from the original 50 million (red, solid), through 5 million (blue, dashed), down to 0.5 million samples (green, dotted). 58

Table of Acronyms

SPL	Sound Pressure Level	6
PWL	Power Level	7
IL	Intensity Level	7
IR	Impulse Response	8
FR	Frequency Response	9
GTD	Geometric Theory of Diffraction	17
UTD	Universal Theory of Diffraction	17
BTM	Biot-Tolstoy-Medwin	19
BT	Biot-Tolstoy	19
PCM	Pulse-code modulation	28
3D	Three-dimensional (space)	36
2D	Two-dimensional (space)	38
BFS	Breadth-First Search	39
MSE	Mean-Square Error	58
HT	Hyper-threading technology	60

Table of Symbols

c	Speed of sound = 340.29 m/s	5
K	Bulk-modulus, or stiffness, of a medium	5
ρ	Density of a medium	5
γ	Adiabatic constant of a medium	5
P	Pressure of a medium	5
$\bar{R}$	Ideal gas constant	5
V	Volume of a gas	5
T	Temperature of a gas	5
M	Molar mass of a gas	5
p	Sound pressure	6
Φ	Sound power	6
I	Sound intensity	6
A	Area of incident sound	6
$h(t)$	Impulse response function	8
$H(\omega)$	Frequency response function	9
$\mathcal{F}\{f\}$	Fourier transform	9
φ	Displacement potential	20
$\mathbb{H}(t)$	Heaviside step function	20
β	Biot-Tolstoy beta function	20
S	The source	21
R	The receiver	21
z_{min}	The coordinate (in wedge coordinates) of the lower extent of the wedge	21
z_{max}	The coordinate (in wedge coordinates) of the upper extent of the wedge	21
l_S	Distance from the source to the point of diffraction	21
l_R	Distance from the reciever to the point of diffraction	21
r_S	Radius from the source to the wedge	21
r_R	Radius from the receiver to the wedge	21

z_S	The coordinate (in wedge coordinates) of the nearest point to the source on the wedge	21
z_R	The coordinate (in wedge coordinates) of the nearest point to the receiver on the wedge	21
ϕ	The angle which the shortest path makes with respect to the direction of the wedge	21
θ_w	Angle of the wedge	22
θ_S	Wedge-to-source angle	22
θ_R	Wedge-to-receiver angle	22
D	The directivity function of a secondary edge-source	25
q	The source signal function	25
v	The wedge coefficient $= \pi/\theta_w$	27
f	Sampling frequency of the discrete IR function	33
S'	Mirrored source	54
R'	Mirrored receiver	54

Chapter 1

Introduction

The development of realistic rendering frameworks has become one of the faces of the advancement of computer technology. Post-processing, special-effects and animation techniques have been completely revolutionized over the past couple of decades. What once consisted of painstaking construction of physical environments, creative application of make-up, and careful planning of stunts has been augmented with safer, adjustable, and convincing computer simulations.

The benefits brought by realistic rendering methods are not limited to the entertainment industry. In the architectural field, computer simulations allow complete walkthroughs of structures that are yet to be constructed [21]. In the biomedical field, interactive rendering techniques may allow noninvasive examination of organs and other biological structures by doctors and scientists [14].

Recently, realistic acoustic simulations have also seen similar advances. Sophisticated algorithms for sound synthesis [15] [29] [54] are providing an alternative to the traditional methods of capturing sounds and providing automated, believable sound effects. Other works in the field aim to speed up acoustic simulation to bring convincing and immersive environments into real-time applications such as interactive walkthroughs and computer gaming [21] [48]. Despite this ongoing research, many current acoustic simulation systems still do not accurately model some properties of sound propagation. Driven by demand for higher quality and more elaborate visuals, the priority for research and development has often been given to visual simulation. Additionally, the accurate modelling of acoustic propagation is a considerably difficult problem covering a wide range of frequency bands in comparison to the three bands (red, green, and blue) which are often sufficient for visual simulation [40].

Nevertheless, we remark that sound is part of the immersive experience. Current research on the physiological impact of audio demonstrates how important the sense of hearing is to us. It has been shown that not only do proper audio cues assist viewers in localizing objects, but also improve the apparent quality of graphical environments [9] [39]. The presence of sound can even substitute for poorer visual performance, while maintaining the same perceived experience [23]. As visuals are quickly becoming indiscernible from reality, the accurate simulation of audio becomes increasingly important to reach the next level of realism in virtual environments.

Many current acoustic propagation systems use ray optics techniques (also called geometrical techniques), such as ray tracing [49], beam tracing [11], or the image source method [1]. Ray tracing is particularly a popular technique in more recent acoustic models due its accuracy and flexibility. Additionally, as many existing visual simulation systems already incorporate ray tracing, it can be relatively easy to extend the system to acoustic simulation. However, many of these systems, originally intended for visual applications, fail to account for some properties of sound. Diffraction, a primary mode of sound scattering and transport, is one such property that can be difficult to simulate in these systems. Although there are several works which attempt to simulate diffraction [12] [49], these approaches often utilize the Universal Theory of Diffraction (UTD) [18] which makes a high-frequency, infinite edge assumption (see Section 3.3.1). As the size of an acoustic wave is often comparable to the size of objects in the environment, such an assumption leads to large errors in the simulated results, notably at lower frequencies [43].

As an alternative to the use of ray optics techniques, many acoustic simulation systems use numerical methods. These methods employ volumetric discretized representation of the scene and attempt to solve the wave equation directly [35]. Hence, numerical acoustics methods are able to account for all wave phenomena, including diffraction. However, this representation is very resource intensive. Numerical methods are considerably slower than ray optics methods, notably in larger environments.

From an artistic perspective, it is often advantageous to provide control mechanisms to allow animators or sound engineers to achieve a desired effect. Both ray tracing and numerical simulations can provide accurate and detailed simulation solutions for complex environments. However, in the event that an animator wishes to alter a given solution, it can be difficult to determine which parts of the environment are associated with the different components of the final result. This is particularly the case in non-visual situations, which lack the spatial relationships between the simulated result and the initial 3D representation of the environment.

In this work, we address some of the aforementioned shortcomings in accuracy, per-

formance, and artistic control faced by existing acoustic simulation systems. Our contributions are as follows. We provide a model of diffraction that produces a time-domain solution to acoustic propagation accurate at all wavelengths. Our approach also allows the user to easily determine the parts of the environment which contribute to the different parts of the simulation result. Finally, we provide a technique that can be seamlessly integrated into existing rendering solutions using ray tracing frameworks.

This thesis is organized as follows. We begin with a brief overview of the principles and physics of sound in Chapter 2. We then look over the current state of the art in the graphics and acoustic simulation fields in Chapter 3. In Chapters 4, we further investigate the Biot-Tolstoy-Medwin formulation of diffraction, from which this work is derived. Then, in Chapters 5 and 6, we describe our approach to simulate diffraction and the implementation of our algorithm, respectively. In Chapter 7, we present our results and compare them against the expected results available in the literature. Finally, we outline directions for future work and conclude in Chapter 8.

Chapter 2

Background

As this work deals deeply with acoustics, this section aims to provide the reader with an introductory background on some of the principles governing the behaviour of sound as well as some basic terminology that is used in this work.

This chapter begins with an introduction into the principles of sound in Section 2.1. Then, as our work focuses specifically in modelling the pheonomenon of diffraction, in Section 2.2 we provide an introduction into modelling this phenomenon itself with the tools from the previous section. Finally, for completeness, the chapter concludes with an overview of ray tracing in Section 2.3.

2.1 Principles of Sound

Sound is the name of the phenomenon that we are able to detect through our sense of hearing. Although our auditory system is complex and can distinguish a large variety of different sounds, in the purely physical sense, sound can be described as a disturbance in the air that surrounds us. This disturbance travels through the medium by means of longitudinal waves. The waves are emitted by a source which mechanically moves the medium. These waves then traverse the medium, interacting with other objects within the medium until they finally are detected by a receiver. In the case of our biological bodies, the intricate components of the auditory canal within our ear pick up these disturbances and convert them to neural signals which are then interpreted by our brains.

Since sound is a wave by nature, we can decompose any sound into a combination of waves of different frequencies which comprise the sound. We can, hence, talk about

the physics of sound in terms of distinct waves. Each wave can be described by two key properties: its frequency (measured in Hz), and its amplitude (often measured in dB). These two properties are discussed further below. Our ears are capable of discerning sounds with a frequency as low as 20 Hz, and as high as 20 kHz. Sound, however, is by no means restricted to this range.

This section is intended to provide a brief introduction to some principles of sound that are relevant to this work. It is by no means intended to be comprehensive. Interested readers should refer to a more complete work on acoustics such as the work by Rienstra and Hirschberg [38].

2.1.1 Speed of Sound

The speed of sound generally refers to the speed at which a sound wave travels in the air that surrounds us. It is often treated as a constant like the speed of light. However, in reality, the speed of sound – or more generally, the speed of any arbitrary longitudinal wave in an medium – is variable and depends on the properties and the state of the medium. Specifically, if we were given the bulk-modulus (or stiffness) of the medium, K, and the density of the medium, ρ, the speed of sound, c, in that medium would be given by the Newton-Laplace equation,

$$c = \sqrt{\frac{K}{\rho}}. \tag{2.1}$$

Note that c is often also used as the universal constant for the speed of light. However, in this work, we will use this symbol to refer to the speed of sound.

Since $K = \gamma P$, where γ is the adiabatic constant and P is the pressure of the fluid, alternatively we have

$$c = \sqrt{\frac{\gamma P}{\rho}}. \tag{2.2}$$

For an ideal gas, we have the relation $PV = n\bar{R}T$, where V is the volume of the gas, n is the number of moles, $\bar{R}$ is the universal gas constant, and T is the temperature of the gas. Combining these two equations and the further observation that $\rho = nM/V$, where M is the molar-mass, we have

$$c = \sqrt{\frac{\gamma n \bar{R} T}{\rho V}} = \sqrt{\frac{\gamma \bar{R} T}{M}}. \tag{2.3}$$

We observe by 2.2 that

$$c \propto \sqrt{P}, \tag{2.4}$$

and by 2.3 that

$$c \propto \sqrt{T}. \tag{2.5}$$

Since, in this work, we will be working with sound waves that we hear, we will be using the speed of sound in the air. We note that while air is not an ideal gas the above relation still provides us of a good approximation of the speed of sound in the air. Furthermore, since the speed of sound is function of both air temperature and pressure, we assume a temperature of 15°C and a pressure of 1.0 atm. Under these conditions the speed of sound is measured to be $c = 340.29$m/s. For the remainder of this work, any numerical calculations involving c will be performed using this value.

2.1.2 Amplitude of Sound

There are several methods of quantifying the amplitude or strength of a sound. Acoustic waves propagate by means of local deviations from the nominal pressure of the medium. The deviations in pressure are measurable and are usually given by the symbol, p, enumerated in pascals (Pa).

The human ear is capable of detecting sounds with a pressure differential as small as 20 μPa for sounds at a frequency of 1 kHz. Accordingly, this is referred to as the threshold of human hearing. While there is no specific upper threshold of human hearing, given earth's atmospheric pressure, sounds with differentials as high as 101 kPa are possible with no distortion. Since this range of pressures covers several orders of magnitudes, sound pressure is instead often measured in the logarithmic decibel scale (dB).

As the decibel scale is a relative scale, it depends on a reference level to represent 0 dB, p_{ref}. Several reference levels are common so one must take care to verify the reference level used for the specific measurement. For example, when the threshold of human hearing (20

μPa) is used as the reference pressure, the scale is referred to as *Sound Pressure Level* or SPL. Hence,

$$\mathrm{SPL} = 20 \log_{10} \left(\frac{p_{\mathrm{rms}}}{20\mu\mathrm{Pa}} \right), \tag{2.6}$$

where p_{rms} is a root mean square measurement of the pressure of the signal.

Alternatively, we can measure the total power, Φ, produced by an acoustic signal. In this case, the power is measured in Watts and the reference level used is 10^{-12} W. This quantity is generally referred to as the *Power Level* or PWL. Hence,

$$\mathrm{PWL} = 10 \log_{10} \left(\frac{\Phi}{10^{-12}\mathrm{W}} \right). \tag{2.7}$$

Since it may be difficult to measure the total power level of a sound after propagation, instead it may be more useful to measure the intensity, I, of the sound, where intensity is power per unit area or

$$I = \frac{\Phi}{A}. \tag{2.8}$$

In this case, we can measure the *Intensity Level* (IL) of the sound, which uses 10^{-12} W/m^2 as the reference level:

$$\mathrm{IL} = 10 \log_{10} \left(\frac{I}{10^{-12}\mathrm{W/m^2}} \right). \tag{2.9}$$

Note that the aforementioned series of measurements by no means constitutes an exhaustive list of possible measurements or possible reference levels with respect to audio strength. For example, some audio equipment uses the level 0 dB to represent the maximum level that can be recorded by the equipment without distortion. Accordingly, on these devices all sound levels would have a negative value under normal operating conditions.

In this work, since we are dealing with sound propagation, we will be working with acoustic pressure, p, and use 0 dB to represent the nominal level of the source audio signal. Positive levels will thus indicate a strengthening of the signal, while negative levels will indicate an attenuation of the signal.

2.1.3 Diffraction

Diffraction is the name given to a phenomenon which occurs when waves interact with small objects or small openings. More specifically, as a wave approaches a small object, it tends to bend around the object, and as a wave passes through a small opening, it tends to spread out after passing through the opening. Being a wave phenomenon, sound exhibits these same behaviours when interacting with objects in the environment. It is for this reason, for example, that one can hear the sounds originating from behind a physical barrier.

The ubiquity of this mode of acoustic propagation makes it an important cue in understanding our environments, as well as making it an important method of communication in situations where line-of-sight is not always present.

Diffraction also alters the characteristics of the signal being diffracted, typically by attenuating higher-frequency signals. In addition, diffraction may amplify or even nullify signals in certain cases due to interference between different paths of a diffracted signal.

Diffraction, as a wave effect, is common to other forms of wave-based energy propagation as well (e.g., light, ocean, and earthquake waves). However, the significance of diffraction in the overall behaviour of these other forms of energy transfer can vary considerably. For example, most effects of the diffraction of visible light can only be observed in the microscopic scale while, on the other hand, low frequency bass sounds can readily diffract around large obstacles. This variation is due to the fact that the effect of diffraction is dependent on both the wavelength of the wave and the size of the obstacle from which it diffracts. With a larger ratio of wavelength to obstacle, we have a stronger deflection. In the case of light waves, this ratio is considerably small, which prevents us from seeing around obstacles.

2.1.4 Impulse Response

In acoustic modelling or signal processing, the impulse response (IR) describes the transmission properties of any energy transmission system. Specifically, the IR measures the time-variant output of the system when a brief, ideally infinitely narrow, pulse is applied to the system [30]. If we have a measure of the impulse response of a system, the output of the system given an input signal can be predicted by convoluting the input with the IR. Mathematically, given the signal and the IR, denoted by $S(t)$ and $h(t)$ respectively, the output, denoted by $o(t)$, is given by the integral

$$o(t) = \int_{-\infty}^{\infty} h(\tau)S(t-\tau)d\tau. \tag{2.10}$$

Hence, the IR provides a convenient method to quantify the behaviour of the system so that it needs to be measured only once in order to process any arbitrary signal. Conversely, knowing the IR of the system also allows one to reconstruct the original input signal given a measured output from the system.

In practical terms, the IR serves to represent the effects that seem to be applied to sounds when they are heard in certain environments. For example, sounds heard in long corridors or cathedrals have distinctive qualities that allow us to quickly discern the recording location just by listening to the recording. These effects, applied to the sound by the environment, are often colloquially referred to as *room effects* [31].

The IR is, therefore, an important measure for determining the characteristic sound of an environment. Its simulation can be used to emulate how sounds would be heard in a given environment. Alternatively, the simulation could be used to design environments that have (or that avoid) certain acoustical properties.

2.1.5 Frequency Response

The frequency response, or FR, is a measure of the resultant behaviour of an energy transmission system when it is exposed to a signal. The FR measures the attenuation of the different frequency bands of a signal as the signal passes through the system. Hence, it can be represented as a frequency versus gain (or loss) graph. The FR is related to the IR in that the FR is the Fourier transform of the IR [33]:

$$H(\omega) = \mathcal{F}\{h(t)\}, \tag{2.11}$$

where $H(\omega)$ denotes the frequency response and $\mathcal{F}\{f\}$ denotes the Fourier transform of a given function.

2.2 Diffractive Modelling

Diffractive modelling is the computational simulation of diffraction. In this work, we use it as a basis for generating an estimate of the IR of an environment (as described in

Section 2.1.4). Conceptually, the method of simulating the IR can be thought of as issuing a pulse at a given sound source. The pulse interacts with the environment and is measured at the receiver. The measured result is then taken to be the IR.

The most basic case in diffraction modelling is the case of diffraction against an infinite wedge. When sound diffracts over an infinitely long wedge, the IR typically takes the form of a heavy-tail exponential fall-off. This is due to the fact that acoustic energy reaches the receiver first and with greatest strength by route of the apex. However, since the signal is a pulse, this initial path conveys the first instantaneous reception of energy. The receiver then continues to register a signal due to the reception of the pulse from other paths of diffraction along the edge.

As the time of reception is related to the path length, the contributing paths must be incrementally increasing in length. Hence, following the initial path by the apex, the contributing path would be the next shortest path, with a diffraction point adjacent to the apex. It so happens that there are two such points (one on either side). Accordingly, we find two (upper and lower) symmetric paths of equal length, $l_{\text{upper},S} + l_{\text{upper},R} = l_{\text{lower},S} + l_{\text{lower},R}$ (see Figure 4.1). As time progresses, the contributing subordinate paths are of increasing length, and thus face greater attenuation.

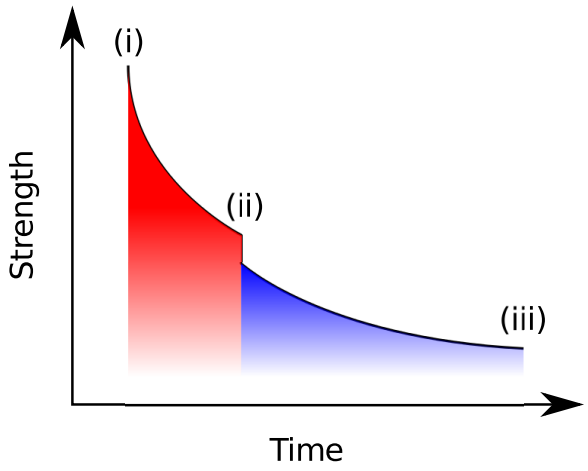

Figure 2.1: Depiction of the three phases of the IR of a finite edge diffraction event: (i) indicates the initial phase where the signal is diffracted by both upper and lower paths, (ii) indicates the half-strength phase where the signal is diffracted by only upper or only lower paths, and (iii) indicates the final phase where both the upper and lower ends are reached and no further contributions occur.

On the finite wedge, however, Svensson et al. [43] notes that the IR has three distinct

phases. The phases are depicted in Figure 2.1 and described below:

(i) In the initial phase, the IR appears to match the IR of an infinite wedge. The signal reaches the receiver by means of the apex as in the infinite (basic) case. Similarly, the signal continues by means of upper and lower contributing paths.

(ii) In the next phase, the IR eventually suffers an instantaneous partial loss of strength. The exponential falloff trend continues thereafter at the lower strength level. This loss is a result of one of the two partial contributing paths reaching an edge of the wedge. For the purpose of illustrating this phenomenon, we assume that the lower path has reached an end. However, we remark that due to the symmetry of the problem, the discussion applies equally to the alternate case.

Once the lower end is reached, there are no further points that can diffract the signal toward the receiver, and contributions by the lower path will cease. Accordingly, we see a fall in the strength of the IR. To characterize this loss in strength, Svensson et al. [43] notes two values originally suggested by Medwin et al. In one work, Medwin [24] suggests that the loss factor is a constant of 1/2. However, in a later work [25], the factor is instead suggested to be given by

$$\frac{l_{\text{upper},S} l_{\text{upper},R}}{l_{\text{lower},S} l_{\text{lower},R} + l_{\text{upper},S} l_{\text{upper},R}}. \tag{2.12}$$

(iii) Ultimately, the upper end of the wedge is reached as well. At this point, contributions from the remaining branch cease, and no more energy is received by diffraction at the receiver.

2.3 Ray Tracing

Ray tracing is an algorithm which is typically used to generate photo-realistic images of a geometrically defined scene through computer simulation. Ray tracing is performed by probing the geometry through the casting of rays. The intersection of the rays with geometrical primitives are used as a basis for higher level interpretations.

2.3.1 Visual Ray Tracing

In visual ray tracing, the rays depict the transmission of light rays through the geometry. Intuitively, we know that we see objects only because light has travelled from some light

source, through the world around us, and into our eyes. The goal of a visual ray tracer is, therefore, to find and simulate the path by which light takes from a simulated light source in the geometric representation of the scene to the observer's eye or *camera*.

Light rarely takes a direct path from the light source to the viewer. Instead, it interacts with objects in the environment before reaching the viewer. These interactions are critical to the operation of the ray tracer. Hence, in defining the light-transport model, the equations that govern how light interactions with a surface, the BSSRDF (bidirectional surface-scattering reflectance distribution function), forms a key concept of the system [32]. Ideally, the BSSRDF captures the process of light striking a material, entering the material, being scattered within the material, and exiting the material at a different point with a different direction. However, due to the complexity of such a computation, the BSSRDF is normally reduced into two approximative functions, the BRDF (bidirectional reflectance distribution function) and the BTDF (bidirectional transmittance distribution function) [32]. These two functions comprehensively cover the cases of reflection from the surface (including diffuse, specular, and mirror) and transmission through the surface (including diffuse and refractive) respectively.

In typical ray tracing applications, the ray is traced starting from the camera (receiver). This may seem counterintuitive, but is done intentionally since the most significant segments of the light path to our perceived appearance of the scene are the final few steps. In this way, each cast ray, then may represent a pixel on the screen. Furthermore, the first object in the scene with which the ray intersects, would be the object seen by the viewer.

This is by no means the only approach to perform the computation of the light path. More sophisticated ray tracing algorithms do not necessarily need to follow this pattern or direction of ray tracing. Some ray tracers, where the lighting side of the problem is more important, start the light path from the light source. These ray tracers are sometimes referred to as *ray casters*. Yet other designs may attempt to start the path from both ends (from the source and the camera), and attempt to join the paths in the middle. These designs are named *bidirectional ray tracers* [6].

2.3.2 Acoustic Ray Tracing

Whereas visual ray tracing attempts to simulate the appearance of a scene by simulating the path of light rays through a scene, acoustic ray tracing attempts to simulate the audio qualities of a scene by simulating the path of sound waves through the scene. Since light and sound share many properties, the principles of design between a visual ray tracer and acoustic ray tracer are similar.

Like light, sound also interacts with surfaces by either reflection, transmission, or absorption. In addition, sound reflection and transmission can be described in terms of similar scattering functions. In contrast to visual simulations, where the spatial distribution and the directionality of the scattering is critically important, in acoustic simulation the temporal distribution and the attenuation of the signals become predominant. Furthermore, the propagative modes of waves that do not directly interact with surfaces become predominant. Examples include the direct sound path between the source and receiver, and the diffraction of acoustic waves around obstacles.

Diffraction is not unique to sound waves as light rays are also subject to diffraction. However, since the wavelength of light is so comparatively small to the scale of visual objects in a scene, diffraction events can be regarded as negligible. Sound waves in the frequency range audible by humans have a wavelength of comparable scale to objects in the scene. Hence, diffraction has a significant effect on the propagation of sound.

Since the scattering modes of propagation are common between light and sound and have been externally addressed in the literature [20] [32], this work instead aims to tackle the challenge of diffraction that is more unique to acoustic simulation.

Chapter 3

State of the Art

The computer graphics field has seen numerous works in the area of acoustic simulation. In this section, we briefly review these works, which are often divided into two main sub-areas consisting of sound synthesis and sound propagation. From these sub-areas, we draw the first two sections: Section 3.1 and Section 3.2, respectively. As the modelling of diffraction is a key challenge in this field, we also outline works which focus on dealing with this problem specifically in a third section, Section 3.3. Finally, as acoustic systems are often subject to real-time requirements, we draw a fourth section of works that may fall into one of the previous categories, but share the common goal of performance optimization, in Section 3.4.

3.1 Sound Synthesis

Works on sound synthesis focus on deriving the wave-form for different sound effects. In order for a sound to be heard by our ears, the sound must be produced by a source and propagated through the environment to our ears. Works in this category specifically focus on the first part of this process. Historically, sound synthesis has been side-stepped through the recording of either existing sounds in nature or sounds produced in studios with physical tools or instruments. However, like its analogue in image synthesis, this process is increasingly being replaced through computer modelling.

Many of those works are derived from or inspired by physical simulations of the events that produce the acoustic energy [29], and rely on precomputation of some of the acoustic properties of the object producing the sound [15] [54] in order to conserve simulation time.

We remark that our work deals with the propagation of an existing wave-form. Hence, it falls under the sound propagation sub-area examined next.

3.2 Sound Propagation

A second group of works deal with the propagation of sound from its location of origin to the listener. As sound propagates, phenomena such as reflection, refraction, and interference alter its characteristics. This body of works attempts to tackle this problem using various models. These models can be broadly catagorized into three types: numerical models, geometric (or ray optic) models, and non-physical models.

3.2.1 Numerical Models

The transfer of sound can be represented using partial differential equations. Hence, its propagation can be simulated by solving these equations. Numerical solvers usually come in two varieties. Volumetric solvers work by descretizing the scene into finite cells and recursively computing the value for the equation at each cell until a solution is reached. These solvers can obtain a detailed solution for the propagation of sound to the entire volume with all sound phenomena. However, the computational and memory requirements for this technique grow with the level of detail and volume of the scene. Adaptive descretization [35] and precomputation [36] have been proposed to mitigate these computational requirements.

Surface-based techniques avoid the volumetric representation of the scene, instead relying on computing the surface-to-surface acoustic transfer. Hence, these methods can be considerably faster than their volumetric counterparts, especially in large open scenes with sparse geometry [26]. One such method, the Equivalent Sources Method (ESM) was brought into the computer graphics field by the work of James et al. [15], and it has also found its way into more recent research [26]. However, as the geometric complexity of the scene and the number of surfaces increases, the performance of surface-based techniques quickly begins to suffer.

3.2.2 Geometric Models

The majority of models used in acoustical propagation employ geometric acoustics techniques. These techniques follow the same set of principles that govern ray optics in visual

rendering. One technique extensively employed in these models is ray tracing (see also Section 2.3). This technique has become one of the staples of photo-realistic rendering in the computer graphics field, and it has found its place in acoustical simulation as well [19]. Ray tracing techniques have become quite popular due to their flexibility in handling arbitrary geometry and complex scenes. Other techniques include beam tracing [11] and the image-source method [1].

Despite their flexibility, many ray optics models neglect the simulation of diffraction. This is due, in part, to the fact that some of these models originated from other fields where diffraction is not a significant effect. For example, as outlined in Section 2.1.3, in photo-realistic rendering the wavelength of light is several orders of magnitude smaller than the objects in the scene. In acoustical simulation, however, the wavelength of sound can be comparable or even larger than objects in a scene. Hence, neglecting diffraction can have significant impact on the simulation accuracy. We discuss the simulation of diffraction in geometrical methods in Section 3.3.

3.2.3 Non-Physical Models

Wölfle and Landstorfer [53] described a model using neural network techniques to simulate wave propagation along what they term as dominant paths. In their model, the dominant paths simply consider the rooms in which a wave travels and ignore all time-varying or non-permanent geometry (for example, doors and soft obstacles). This model was originally designed for the simulation of radio wave propagation. However, we remark that radio waves and sound waves have comparable wavelengths, and thus comparable behaviour. Accordingly, this technique could seamlessly be extended to the acoustic domain. While their model does not simulate diffraction directly, its effects are accounted for through the neural network learning scheme. However, the network requires training prior to use, and it cannot model specific diffraction effects.

3.3 Diffractive Modelling

Diffraction, as a wave phenomenon, does not have a simple geometric interpretation. As diffraction occurs around edges of objects, it can be difficult to integrate and quantify in systems that operate on the assumption that rays will interact with large planar surfaces (as is the case for the simulation of reflection or refraction). For these reasons, simple statistical ray tracing methods are unable to capture the effects of diffraction, although several approaches have been suggested to this specific challenge [16] [41].

Further compounding the issue, the outcome of a diffraction event on a signal is dependent on the ratio of the wavelength to the size of the object. Hence, taking a localized computation of diffraction is not possible. Beam tracing [21] [49] or other ray tracing techniques, such as sonel mapping [16], can assist in finding the diffractive interfaces. However, the local edge diffraction computation remains complex as the outcome is dependent on the relative sizes of the obstacle and the wavelength.

3.3.1 Universal Theory of Diffraction

In the case where the wavelength is much smaller than the length of the edge on which diffraction occurs, the computation of diffraction can be greatly simplified. Taking advantage of this observation, Keller [17] made a "high-frequency, relatively infinite edge" assumption. By leveraging this assumption, Keller devised an efficient formulation of diffraction known as the Geometrical Theory of Diffraction (GTD). This method was further refined by Kouyoumjian and Pathak [18] into what is known as the Uniform Theory of Diffraction (UTD). The GTD and UTD formulations allow the computation of diffraction in a local interaction context similar to the computation of refraction and reflection. Hence, they are very attractive methods to use in geometric models which already incorporate the aforementioned phenomena. Consequently, they have seen widespread use [44] [49].

The high-frequency, relatively infinite edge assumption, however, may not always be suitable for acoustic simulation. Audible sound-waves have frequencies as low as 20 Hz and wavelengths exceeding 15 m. These lengths are considerably larger than many household obstacles and thus violate the assumptions of this method. Hence, works using UTD suffer increased error in lower frequency signals.

3.3.2 Biot-Tolstoy-Medwin

Biot and Tolstoy [3] developed an alternative formulation for diffraction that does not make the assumptions made by the UTD. The original formulation by Biot and Tolstoy provides a solution for infinite wedges. Medwin et al. [25] and later Svensson et al. [43] extended the Biot-Tolstoy model to finite wedges in addition to providing an interpretation of the model for multiple diffraction. As BTM does not make the relatively infinite edge assumption, it can provide accurate results for the entire frequency domain. Accordingly, this method has been used as a reference standard with which to evaluate other methods [26]. Our work builds upon these results, and we describe this method in further detail in Chapter 4.

3.4 Real-Time Computation

Many applications of acoustic simulation have real-time constraints such as in interactive applications. In these cases, simulation results must be produced quickly, and computational power must be shared with other real-time requirements (e.g. graphics, AI). Accordingly, several works in acoustic propagation aim at optimizing existing techniques. Recent works include the work by Tsingos et al. [48] in using the GPU to assist in the evaluation of the sound scattering integral. Lauterbach et al. [21] demonstrated an efficient algorithm using a method entitled "frustum tracing", which is similar to beam tracing, using multicore computing and SIMD enhancements. More recently, Raghuvanshi et al. [36] was able to bridge the real-time gap with numerical acoustics methods. This was accomplished through the use of precomputation as well as generalizations for higher frequencies and for the later stages of reverberation (resonance or echoing of sound).

Chapter 4

The Biot-Tolsoy-Medwin Formulation

Our method to compute the diffractive IR builds upon the model developed originally by M. A. Biot and I. Tolstoy [3], extended by Medwin et al. [24] [25], and further extended by Svensson et al. [43]. In order to gain an understanding of its basis and how this model is applied to current applications, this chapter will discuss its formulation and the extensions which led to its design up to and including the work by Svensson et al. [43]. We begin by discussing the original model by Biot and Tolstoy in Section 4.1. We then look at the enhancements added by Medwin et al. [24] [25] in Section 4.2 and Svensson et al. [43] in Sections 4.3, respectively. Some practical issues in the numerical implementation of these models are examined in Section 4.4. Finally, in Section 4.5, we discuss our extensions and contributions to the model which form the basis of this thesis.

4.1 The Original Model

In 1957, M. A. Biot and I. Tolstoy, by employing the method of normal modes developed a model of wave propagation in cylindrical space (r, θ, z) [3]. The intuition of the method is to determine the effect a wedge has on a wave by modeling the fluid around the wedge. The wave, in this case, is induced by an "instantaneous explosion" in the fluid caused by a sudden injection at the wave source modeled with the Dirac delta function.

In this model, the wedge acts as the boundary of the space, and it is centred with the edge residing at $r = 0$. The edge itself extends infinitely along the z axis, and each of the wedge's two planar surfaces are located $d\theta$ apart and also extend infinitely in z and r. If one were to take a cross-section of the space, therefore, at a fixed z, cutting across the r and θ dimensions, the space would have a form similar to a pie-with-a-missing-slice.

Biot and Tolstoy then use the method of normal modes to develop an analytical solution given any arbitrary wedge angle, and source (injection) and receiver (sampling) positions. This solution is given by Equation 5.24 of [3] and reproduced as follows (with the symbols changed for consistency):

$$\frac{\partial \varphi}{\partial t} = \frac{c}{4\pi\theta_w} \frac{\beta}{r_S r_R \sinh y} e^{-\pi y/\theta_w} \times \mathbb{H}\left(t - \frac{r_S + r_R}{c}\right), \tag{4.1}$$

where φ is the displacement potential of the fluid, $\mathbb{H}(t)$ refers to the Heaviside step function,

$$\beta = \beta_{++} + \beta_{+-} + \beta_{-+} + \beta_{--}, \tag{4.2}$$

$$\beta_{\pm\pm} = \frac{\sin([\pi/\theta_w][\pi \pm \theta_S \pm \theta_R])}{1 - 2e^{-\pi y/\theta_w} \cos([\pi/\theta_w][\pi \pm \theta_S \pm \theta_R]) + e^{-2\pi y/\theta_w}}, \tag{4.3}$$

$$y = \operatorname{arccosh} \frac{c^2 t^2 - (r_S^2 + r_R^2 + dz^2)}{2 r_S r_R}, \tag{4.4}$$

and the remaining parameters are given in Figure 4.1 and Figure 4.2. The symbol $\pm$ is written for simplicity in $\beta_{\pm\pm}$: the two $\pm$'s refer to the two $\pm$'s in the numerator and the denominator of the function. The result is given in terms of the change in displacement potential, denoted by $\partial\varphi/\partial t$.

The mathematics used for the derivation of these equations is outside of the scope of this thesis, but we kindly direct interested readers to the original paper by Biot and Tolstoy [3].

4.2 Subsequent Enhancements

While Biot and Tolsoy's formulation of diffraction provided a comprehensive mathematical model of diffraction over infinite wedges, their model left some questions unanswered. Namely, the BT model does not clearly describe how diffraction across a finite wedge should be interpreted. Furthermore, a wave may diffract over more than one edge, leading to higher order diffraction. Biot and Tolstoy's work addresses diffraction over only one edge. In the early 1980's, as computers were becoming increasingly powerful and commonplace, Medwin sought to extend the BT model with the computational possibilities that these machines created.

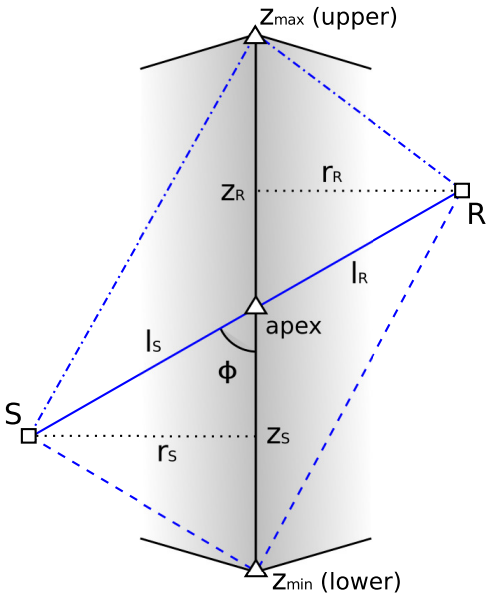

Figure 4.1: A view of a wedge from above. The vertical line indicates the edge of the wedge. Three paths are depicted from the source (S) to the receiver (R): the shortest path (the solid line through the apex), the longest lower path (the dashed line through z_{min}), and the longest upper path (the dash-dotted line through z_{max}). The parameters l_S and l_R describe the distances from the source and receiver respectively to the point of diffraction. The parameters r_S and z_S describe the radius from the edge and the position along the edge of the source, respectively, while the parameters r_R and z_R correspond to the radius from the edge and the position along the edge of the receiver, respectively. Finally, the parameter ϕ describes the angle which the shortest path makes with respect to the direction of the edge.

One fundamental change, however, that Medwin [24] makes to the formulation is in the pressure wavefront simulated by the model. Medwin notes that by using an "explosive, instantaneous injection", the displacement potential is given as

$$\varphi = \frac{1}{4\pi r}\mathbb{H}\left(t - \frac{r_S + r_R}{c}\right). \tag{4.5}$$

Since pressure is given by

$$p = -\rho\frac{\partial^2\varphi}{\partial t^2}, \tag{4.6}$$

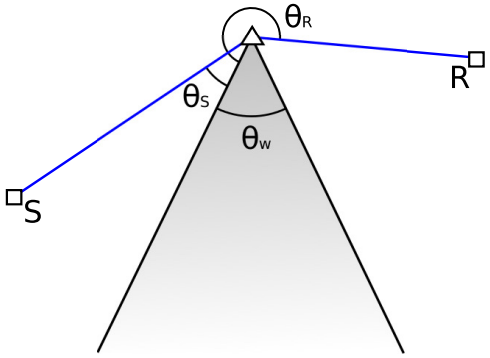

Figure 4.2: A cross section of a wedge from the side. A path is depicted from the source (S) to the receiver (R). The parameter θ_w represents the angle of the wedge, while the parameters θ_S and θ_R represent the wedge-to-source and wedge-to-receiver angles, respectively.

and the second order derivative of $\mathbb{H}(t)$ is $\delta'(t)$, the BT model creates a pressure wave in the form of the derivative of the Dirac delta function, consisting of an instantaneous compression, followed by an instantaneous decompression.

Instead, Medwin opts to represent the source as producing a constant flow. In this scheme, at the given time the source would begin injecting material at a constant rate and continue injecting material indefinitely. Hence,

$$\frac{\partial \varphi}{\partial t} = \frac{1}{4\pi r}\mathbb{H}\left(t - \frac{r_S + r_R}{c}\right). \tag{4.7}$$

In this way, Medwin's formulation produces a pressure wave in the form of the Dirac delta function [24].

As Medwin notes, since the boundary conditions for a rigid wedge are the same for both an instantaneous injection and for a constant flow, the change in the formulation becomes simply a change in notation. Hence, Equation 4.1 becomes

$$\frac{\partial^2 \varphi}{\partial t^2} = \frac{c}{4\pi\theta_w}\frac{\beta}{r_S r_R \sinh y} e^{-\pi y/\theta_w} \times \mathbb{H}\left(t - \frac{r_S + r_R}{c}\right). \tag{4.8}$$

Replacing $\mathbb{H}(t)$ with a signal source function $S(t)$ and substituting the result into Equation 4.6 we get

$$p(t) = \frac{-\rho c}{4\pi\theta_w} \frac{\beta}{r_S r_R \sinh y} e^{-\pi y/\theta_w} \times S\left(t - \frac{r_S + r_R}{c}\right). \tag{4.9}$$

The formulation above, with a Dirac delta pressure wave, is generally referred to as the Biot-Tolstoy-Medwin or BTM formulation.

4.2.1 Finite Wedge Diffraction

Although a representation of diffraction from an infinite wedge is useful from a theoretical standpoint, practically, most wedges are in fact finite. Hence, having a formulation for a finite wedge is more useful for real applications. In his 1981 paper, Medwin addresses this issue [24]. Medwin observes that the instantaneous diffraction contribution to the IR can be visualized as the intersection points between the edge of the wedge and the expanding wavefront from the source. For an infinite wedge, once the wavefront has reached the edge, there will always be two points of intersection between the wavefront and the edge. In the case of the exact minimal wavefront which just touches the edge and, hence, has one point of intersection, we treat as two co-located points.

On a finite wedge, however, eventually the spherical wavefront will outgrow the edge. Furthermore, the edge may be offset with respect to the wavefront, and there may never be two points of intersection. Although Medwin identifies two possible configurations, for clarity, we will divide his first configuration into two sub-configurations leading to three cases:

1. The wavefront first reaches the edge in the exact centre. In this case, there will be two points of intersection until the wavefront outgrows the edge. At that point, there will be no points of intersection. See Figure 4.3, Phases 0, 1, 2, and 3.

2. The wavefront first reaches the edge at a point offset from the centre. In this case, there will initially be two points of intersection. However, the wavefront will reach the end of one side of the edge before reaching the end of the other side. Therefore, there will be a period for which the wavefront intersects the edge at only one point. Finally, the wavefront will completely outgrow the edge, at which point there will be no points of intersection. See Figure 4.3, Phases 0, 1, 2, 2b, and 3.

3. The wavefront growth is far from the edge and first reaches the edge at an extrapolation of the edge. In this case, initially the wavefront will not intersect the edge directly. Eventually, the side of the wavefront will reach the edge and there will be

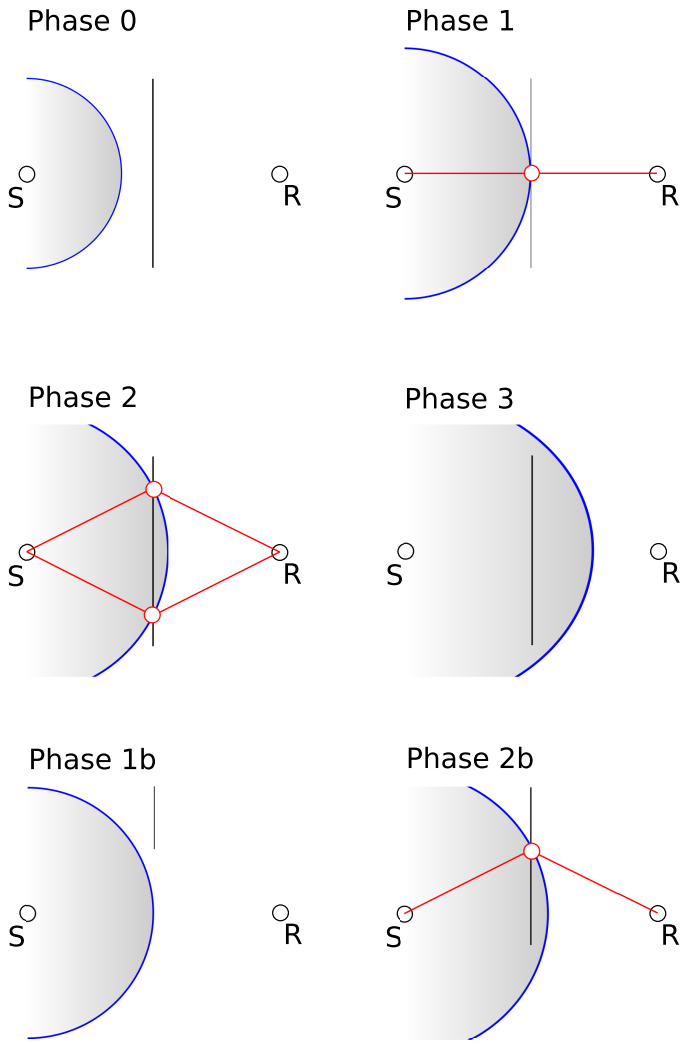

Figure 4.3: The phases of interaction between a wedge and the spherical wavefront emanating from the source (S) towards the receiver position (R). The wave travels toward the edge (Phase 0) and eventually intersects the edge at one point (Phase 1), causing one initial diffraction path to the receiver. As the wavefront grows, it intersects the edge twice (Phase 2), causing *upper* and *lower* diffraction paths. Finally, the wavefront will outgrow a finite edge terminating the intersections (Phase 3). In the case of an offset edge, there may not be an initial head on intersection (Phase 1b). Alternatively, there may be a period where the wavefront has outgrown the edge on one end, but not on the other (Phase 2b), leading to only an upper or a lower diffraction path.

one point of intersection. Finally, the wavefront will outgrow the edge and there will be no points of intersection. See Figure 4.3, Phases 0, 1b, 2b, and 3.

To produce the result for a finite wedge, then, Medwin accordingly scales the result given by Equation 4.9 by 2/2, 1/2, or 0/2 depending on whether at time t, the wavefront is intersecting the edge, twice, once, or not at all, respectively. It is on the basis of this observation that later Svensson et al. [43] describes the IR in three distinct phases (see Section 2.1.4).

4.2.2 Double Diffraction

Medwin continued to refine his interpretation of the interaction between the spherical wavefront and the wedge, and in a 1982 paper [25], he suggests that the edge behaves as a "phased continuous line-source which radiates energy from the edge". This line of thinking allowed Medwin to take a "Huygen's approach" in solving diffraction, essentially treating the edge as an infinite number of infinitesimal point sources, each triggered when a wavefront reaches them.

This new interpretation of diffraction is significant because it provides an indication of how each portion of the edge contributes to diffraction result, thereby decoupling the geometric dependence of the BTM model on wedges of infinite length. In so doing, we are able to analyze diffraction events with sources and receivers that have more complex geometries using the same simple geometric interpretations.

Using this new approach, Medwin then demonstrated, in the same paper, how double-diffraction (diffraction over two edges) can be easily represented and computed by dividing the first edge into n secondary sources that would act as the wave source to the second edge. By determining the values for the n secondary sources on the first edge, and performing n integrations from the secondary sources to the receiver and combining the results, diffraction over two edges is, hence, solved. This process is depicted in Figure 4.4.

4.3 The BTM Line Integral

The next major step in the evolution of the BTM formulation was carried out by Svensson et al. [43]. Svensson et al. took Medwin's idea of using a "discrete Huygen's approach" and pushed it one step further: by using a Huygen's approach and describing an edge as a number of secondary wave sources, one should be able to find a solution to integrating

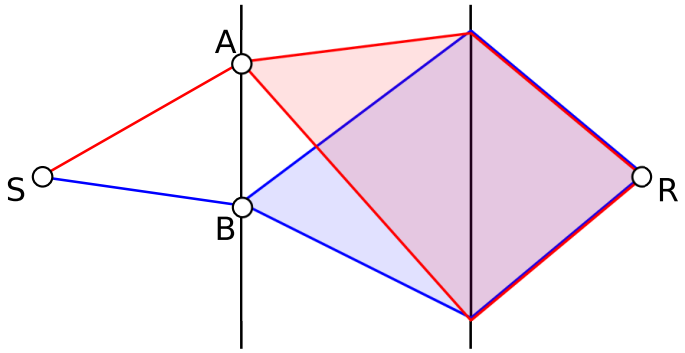

Figure 4.4: The method of simulating double diffraction detailed by Medwin et al. [25]. Several points are selected on the first edge (depicted as A and B) and used as secondary sources in a single diffraction computation with the receiver (R). The aggregate result is then combined with the attenuation of the path from the source (S) to the secondary source.

across the secondary sources along the edge of the wedge, rather than integrating over time.

Note that under the notation by Svensson et al. [43], diffraction is modeled as an IR function, $h(t)$, which has the following relation to the pressure function, $p(t)$:

$$p(t) = \frac{h(t)}{4\pi}. \tag{4.10}$$

In order to find this solution, Svensson et al. started by assuming the existence of a directivity function, $D(\phi_S, \phi_R, \theta_S, \theta_R)$, to represent the energy transfer by the secondary edge source. The directivity function is assumed to be a function of only the incoming and outgoing directions of the diffracted energy and so may be computed independently of the distance of propagation from the primary source, l_S, and to the receiver, l_R. Hence, given a source signal, $S(t)$, the contribution from each secondary source can be described with the following relation:

$$dh(t) = S\left(t - \frac{l_S + l_R}{c}\right) \times \frac{1}{l_S} \times D(\phi_S, \phi_R, \theta_S, \theta_R) \times \frac{1}{l_R} dz, \tag{4.11}$$

where ϕ_S, ϕ_R, l_S, and l_R are functions of the edge position z (these values may be computed trigonometrically from Figure 4.1) , and $D(\phi_S, \phi_R, \theta_S, \theta_R)$ is the unknown directivity

function that models the attenuation of the path. The solution for the IR due to diffraction is then of the form

$$h(t) = \int_{z_{\min}}^{z_{\max}} S\left(t - \frac{l_S + l_R}{c}\right) \times \frac{D(\phi_S, \phi_R, \theta_S, \theta_R)}{l_S l_R} dz. \tag{4.12}$$

Svensson et al. then shows (in Section I.B of [43]) that the unknown directivity function is in fact

$$D(\phi_S, \phi_R, \theta_S, \theta_R) = -\frac{v\beta}{4\pi}. \tag{4.13}$$

where $v = \pi/\theta_w$.

Hence, the solution for the total diffracted pressure becomes

$$h(t) = \int_{z_{\min}}^{z_{\max}} S\left(t - \frac{l_S + l_R}{c}\right) \times \left[- \frac{v\beta}{4\pi l_S l_R}\right] dz. \tag{4.14}$$

The advantage of this formulation is that it leads itself to a trivial geometric interpretation. For example, since each secondary source reradiates the incoming energy according to the directivity function, second-order diffraction can simply be interpreted as a double reradiation with two directivity functions: one for the first wedge and one for the second wedge. Mathematically, this can be expressed as follows:

$$\begin{aligned} dh_2(t) = S\left(t - \tfrac{l_{S,1}+l_{1,2}+l_{2,R}}{c}\right) \quad & \times \quad \tfrac{1}{l_{S,1}} \times D(\phi_{1,S}, \phi_{1,2}, \theta_{1,S}, \theta_{1,2}) \\ & \times \quad \tfrac{1}{l_{1,2}} \times D(\phi_{2,1}, \phi_{2,S}, \theta_{2,1}, \theta_{2,S}) \\ & \times \quad \tfrac{1}{l_{2,R}} dz_1 dz_2, \end{aligned} \tag{4.15}$$

where, $\phi_{1,S}$, $\phi_{1,2}$, $\phi_{2,1}$, $l_{S,1}$, and $l_{1,2}$ are functions of dz_1, and $\phi_{1,2}$, $\phi_{2,1}$, $\phi_{2,R}$, $l_{1,2}$, and $l_{2,R}$ are functions of dz_2.

4.4 Practical Considerations

Although the line-integral formulation proposed by Svensson et al. [43] provides a nice geometric interpretation of the BTM method, it lends itself to some practical implementation

issues that must be solved. While the formulation provides an elegant symbolic solution in the infinite domain, there are some complications which must be overcome in the finite, discrete computation domain.

Audio signals are typically represented in pulse-code modulation (PCM), which samples the pressure of the sound signal many times a second (usually at a rate of 40 kHz or higher). Measuring sound in this format is advantageous because it lends itself to easy manipulation with mathematical primitives. For example, two signals can be easily accumulated through summation, or level adjusted through multiplication. Additionally, a sampled function can easily be converted into the frequency-domain through the use of a Discrete Fourier Transform.

In simulating the diffractive IR through the method proposed by Svensson et al., however, there is a question of how the integration should be sampled. Given that the output function will be sampled at a given rate in the time-domain, how do we subdivide the integration domain (which integrates over the edge length)? Moreover, once we subdivide and compute the integration, how do we map those results to samples in the time domain?

Calamia and Svensson [5] describe two possible strategies for subdividing the integration domain for discrete sampling: sample-aligned segments and even-sized segments.

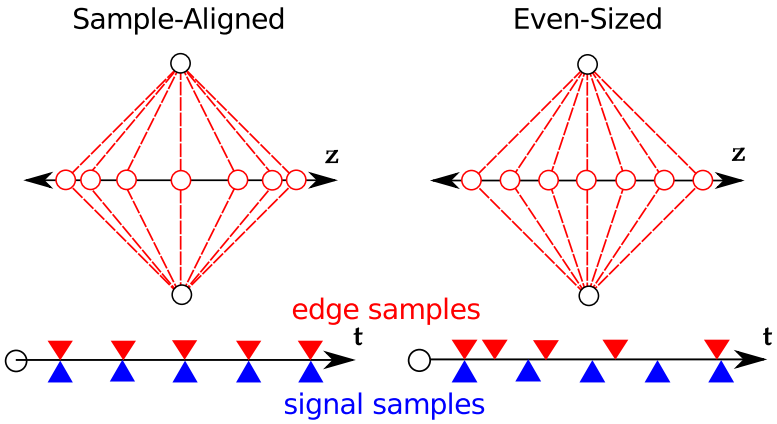

Figure 4.5: Visual depiction of sample-aligned segments (left) versus even-sized segments (right). The upper part of the figure depicts the subdivision of the edge, where z indicates the edge position, while the lower part depicts the temporal alignment of the sampled edge-segments with the discretization of the signal function, where t indicates time.

4.4.1 Sample-Aligned Segments

Under this strategy, the length of the edge is divided into segments of varying length such that the contribution time of each segment is aligned perfectly with the sample periods in the generated IR function. Since each sample is associated with a fixed time, they are also associated by a fixed path length (since the speed of sound is constant). For any given fixed-length path with one deflection, the possible positions of the deflection can be described with an ellipsoid, where the origin and end of the path form the two foci. The bounds of each sample-aligned segment can, thus, be found by intersecting the edge with the appropriate ellipsoid for each sample-aligned segment.

The edge will intersect each ellipsoid once, twice, or not at all similarly to as described in Section 4.2.1. Each ellipsoid represents an instant in time forming the boundary of two samples. The intersection of the ellipsoid then represents the boundary of two sample-aligned segments of the edge. For each sample bounded by the ellipsoids for time i/f and $(i+1)/f$, where f is the sampling frequency, we have two segments defined by coordinates $z_{i,1}$ to $z_{i+1,1}$ and $z_{i,2}$ to $z_{i+1,2}$. The simulated discrete IR function is then simply given by

$$h(i) = \int_{z_{i,1}}^{z_{i+1,1}} -\frac{v}{4\pi}\frac{\beta(z)}{m(z)l(z)}dz + \int_{z_{i,2}}^{z_{i+1,2}} -\frac{v}{4\pi}\frac{\beta(z)}{m(z)l(z)}dz, \tag{4.16}$$

where i represents the index of the sample.

Calamia and Svensson [5] assert that such sampling is advantageous because the direct relationship between edge-segments and time samples makes the sample integration straightforward. The direct relationship also ensures that the sampling rate of the IR is indicative of the true resolution that the edge was sampled at, which avoids problems such as *aliasing* (see Figure 4.6).

On the other hand, performing the ellipsoid intersections to determine the extents of the edge-segments requires complex arithmatic operations involving finding the roots of a quadratic equation. To make matters worse, these intersections must be repeated multiple times since an intersection must be performed for each sample. Calamia and Svensson [5] also point out that the intersections must also be recomputed if the source or receiver moves. This makes the computation very computationally expensive, especially for time-sensitive applications such as real-time applications.

We further assert that the method of intersecting with an ellipsoid may be used only for one edge (single diffraction). Once two or more edges are present, then for any given path length there are an infinite number of paths of any given length. It may, however, be possible to combine this method with even sized segments into a hybrid method.

4.4.2 Even-Sized Segments

With an even-sized segment strategy, the positions of the segments along an edge are precomputed and even in length. Furthermore, these positions do not need change even as source and receiver positions change. The subdivision is fast, and the integration can also be calculated quickly. Hence, this approach is ideal for applications where computation time is critical.

A simple strategy for accomplishing the subdivision is to define a maximum segment length Δz_{max}. Then, for each edge of a given length L, we chose the minimal n that satisfies the inequality $L/n < \Delta z_{max}$. The edge is, thus, divided into n segments, where each segment j is bounded by the coordinates $L{\times}j/n$ and $L{\times}(j{+}1)/n$. The IR contribution for segment j is then given to be

$$h_j = \int_{L\times j/n}^{L\times(j+1)/n} -\frac{v}{4\pi}\frac{\beta(z)}{l_S(z)l_R(z)}dz. \tag{4.17}$$

As the contributions are given in terms of the segment index j which is related to the segment position, the temporal position of the contribution h_j must be computed. Since the instantaneous temporal position of a contribution at a given edge position z is given by

$$t(z) = \frac{l_S(z) + l_R(z)}{c}, \tag{4.18}$$

the contribution h_j applies over the temporal range

$$\frac{l_S(L\times j/n) + l_R(L\times j/n)}{c} \ldots \frac{l_S(L\times (j+1)/n) + l_R(L\times (j+1)/n)}{c}. \tag{4.19}$$

Mapping this temporal range to a function discretely sampled in the time domain (as the final resultant IR will be) may, however, be complex. This is due to the fact that the time of each sample in the function may not align with the above temporal range. In fact, the mapping between the respective time periods of the samples and the temporal ranges of the above contributions may form an N-to-N relationship. This nontrivial mapping can cause aliasing artifacts (see Figure 4.6), and also means that the resolution of the resultant IR function is not indicative of the resolution of the IR integration.

We, however, remark that this type of approach is suitable for computations with higher order diffraction. In these cases, run time increases exponentially with increasing number

of edges. Suppose that the number of subdivisions is proportional to the length of the edge, and the worst edge length is given by L_{max}. The run time of the algorithm will then be given by $O(L_{max}^E)$, where E is the order of diffraction. Hence, for higher order diffraction, performance will suffer.

4.5 Our Contributions

In light of the deficiencies of both of the above sampling methods, particularly in the area of the sampling of multiple diffraction, we instead propose the use of Monte Carlo methods in sampling the line-integral proposed by Svensson et al. [43]. This is accomplished by choosing n independent sets of E points (one point on each edge) for order-E diffraction. The length and, hence, the time of the path (since time is related to the length by the speed of sound) is then computed and associated with a sample in the IR formula.

By using random sampling, we avoid artifacts such as those of structured sampling (as is present in even-sized sampling), while still retaining the ability to quickly compute positions on the edge. Furthermore, by separating the sampling strategy from the geometry of the problem, we are able to freely adjust the accuracy of the result without consideration for the geometric configuration. Specifically, we can incrementally improve the accuracy of the result by running additional samples. With Monte-Carlo integration, any partial result is, in essence, also a candidate final result. The distinction between a partial result and the final result lies simply in the level of accuracy required. This allows for finer control over the accuracy of the result, or over the computational resources to be devoted to the simulation.

In the remainder of this section, we describe our Monte-Carlo sampling strategy and then proceed to extend this strategy to solving higher-order diffraction problems.

4.5.1 Monte-Carlo Sampling

By applying Monte-Carlo methods for integration [13], we can approximate the integral

$$F = \int_{\Omega} f(x)dx \tag{4.20}$$

with the form

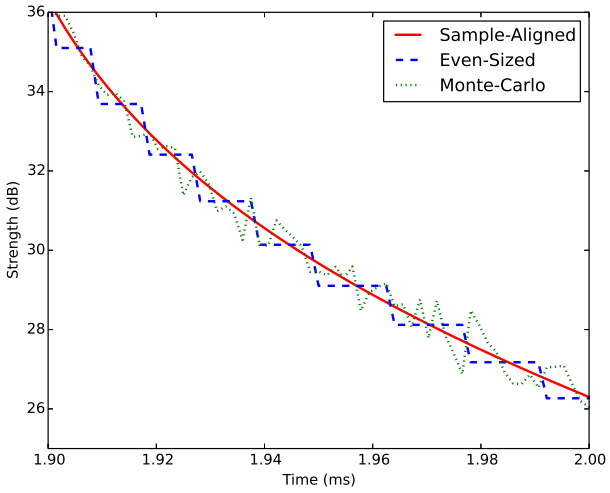

Figure 4.6: A zoomed section of a single-wedge diffraction simulation to compare the error profile of different simulation algorithms. Sample-aligned segment based simulations (solid, red) provide the most accurate result for single diffraction, and serve as a reference in this example. Even-sized segment simulations (dashed, blue) produce a patterned, structural error known as *aliasing*. Finally, Monte-Carlo simulations (dotted, green) produce random error.

$$F \approx \frac{1}{N} \times \left(\int_{\Omega} dx \right) \times \left(\sum_{k=0}^{N} f(x_k) \right), \tag{4.21}$$

where N is the number of samples. From Equation 4.14, we define our derivative to be

$$dh(t,z) = \delta\left(t - \frac{l_S + l_R}{c} \right) \times \left[- \frac{v\beta}{4\pi l_S l_R} \right] dz. \tag{4.22}$$

We use $\delta(t)$ for $S(t)$ since we are measuring the IR. Also from Equation 4.14, our domain is $\Omega = z_{min}..z_{max}$. Hence, we have

$$\int_{\Omega} dz = \int_{z_{min}}^{z_{max}} dz = z_{max} - z_{min} = L. \tag{4.23}$$

It so follows that

$$h(t) \approx \frac{L}{N} \sum_{k=0}^{N} dh(t, z_k), \tag{4.24}$$

where $z_0..z_N$ are randomly chosen points on the edge. Since we have fixed sampling intervals for time, it is useful to rebase the problem in terms of these sample indexes. Hence, we have

$$h(i) \approx \frac{L}{N} \sum_{k=0}^{N} dh(i, z_k), \tag{4.25}$$

where

$$dh(i, z) = D\left(i - \frac{(l_S + l_R) \times f}{c}\right) \times \left[-\frac{v\beta}{4\pi l_S l_R}\right] dz. \tag{4.26}$$

The parameter f in this case represents the sampling frequency of the output. Also note that the quantities l_S and l_R are a function of the edge position z (see Figure 4.1). $D(i) \approx \delta(t)$ such that

$$D(i) = \begin{cases} 1 & : i = 0 \\ 0 & : \text{else} \end{cases} \tag{4.27}$$

Hence, we may alternatively define $dh(i, z)$ to be

$$dh(i, z) = \begin{cases} -\frac{v\beta}{4\pi l_S l_R} & : \frac{i}{f} < \frac{l_S + l_R}{c} < \frac{i+1}{f} \\ 0 & : \text{else} \end{cases} \tag{4.28}$$

We can now approximate $h(i)$ with the following algorithm:

1. Initialize $h(i) = 0$, $i = 0..\inf$.
2. Repeat N times:
   (a) Generate a random coordinate on the edge, z_k.
   (b) Compute $dh(z_k) = -\frac{v\beta}{4\pi l_S l_R}$.
   (c) Compute $i = \lfloor \frac{l_S + l_R}{c} \times f \rfloor$.
   (d) Accumulate $dh(z_k)$ to $h(i)$.
3. Scale $h(i)$, $i = 0..\inf$ by L/N.

4.5.2 Higher-Order Diffraction

Recall from Section 4.3 that using the line-integral formulation, we can trivially define the partial equation for higher-order diffraction. As an example of this application, we derived an equation for second-order diffraction in Equation 4.15. Extending this process to M-order diffraction, we find that the differential equation is of the form

$$dh_M(t) = S\left(t - \frac{l_{S,1} + (\sum_{m=0}^{M-1} l_{m,m+1}) + l_{M,R}}{c}\right) \times \frac{\Pi_{m=0}^{M}(-\frac{V_m \beta_m dz_m}{4\pi})}{l_{S,1} \times (\Pi_{m=0}^{M-1} l_{m,m+1}) \times l_{M,R}}. \quad (4.29)$$

We then apply a modified version of the Monte Carlo approach described in the previous section to solve for $h_M(i)$.

Since $dh_M(t)$ is a derivative on multiple variables $(dz_1..dz_M)$, we revise the domain of integration to be

$$\Omega = z_{1,min}...z_{1,max} \times ... \times z_{M,min}..z_{M,maz}. \quad (4.30)$$

Hence,

$$\int_{\Omega} dz_1..dz_M = \int_{z_{1,min}}^{z_{1,max}} ... \int_{z_{M,min}}^{z_{M,maz}} dz_M...dz_1 = \Pi_{m=0}^{M} L_m. \quad (4.31)$$

It then follows that

$$h_M(t) \approx \frac{\Pi_{m=0}^{M} L_m}{N} \sum_{k=0}^{N} dh_M(t, z_{1,k}, ..., z_{M,k}), \quad (4.32)$$

where $z_1..z_M$ is a randomly chosen set of points, one per wedge to be diffracted along the path, while $z_{m,1}..z_{m,k}$ are k randomly chosen sets of $z_1..z_M$ points.

As we did before, we rebase our problem into sample space. Hence,

$$h_M(i) \approx \frac{\Pi_{m=0}^{M} L_m}{N} \sum_{k=0}^{N} dh_M(i, z_{1,k}, ..., z_{M,k}), \quad (4.33)$$

where

$$dh_M(i, z_1, ..., z_M) = \begin{cases} \frac{\Pi_{m=0}^{M}(-\frac{V_m \beta_m}{4\pi})}{l_{S,1} \times (\Pi_{m=0}^{M-1} l_{m,m+1}) \times l_{M,R}} & : \frac{i}{f} < \frac{l_{S,1} + (\sum_{m=0}^{M-1} l_{m,m+1}) + l_{M,R}}{c} < \frac{i+1}{f} \\ 0 & : \text{else} \end{cases} \quad (4.34)$$

We then rewrite the inner loop of the algorithm to be,

2. Repeat N times:
   (a) Generate a random set of coordinates on the edges of the path, $z_{1,k}..z_{M,k}$.
   (b) Compute $dh_M(z_{1,k}..z_{M,k})$:

$$dh_M(z_{1,k}, ..., z_{M,k}) = \frac{\Pi_{m=0}^{M}(-\frac{V_m \beta_m}{4\pi})}{l_{S,1} \times (\Pi_{m=0}^{M-1} l_{m,m+1}) \times l_{M,R}}. \quad (4.35)$$

   (c) Compute i:

$$i = \lfloor \frac{l_{S,1} + (\sum_{m=0}^{M-1} l_{m,m+1}) + l_{M,R}}{c} \times f \rfloor. \quad (4.36)$$

   (d) Accumulate $dh_M(z_{1,k}..z_{M,k})$ to $h_M(i)$.

Chapter 5

Path Determination

In calculating the multipath interference between the source and receiver, we must find the possible paths that the acoustic waves will take to propagate from the source to the receiver. Since, however, there are an infinite number of paths in which energy can travel, this full computation is intractable. Instead, we simplify the problem in two ways. Firstly, we group similar paths into one path computation. Secondly, we select the paths which have the greatest contribution or significance to the resultant signal. We accomplish this by defining the notion of a *dominant path*. This idea will be explained below in Section 5.1.

In defining these dominant paths in terms of their significance to the final result, we must assume that this significance can be measured such that the paths may be compared. By redefining this notion of significance into a heuristic for shortness, we then reduce the problem of identifying dominant paths to finding the shortest, obstacle-free path in 3D from the source to the receiver. As shortest path search has its own set of difficulties in 3D, we dedicate Section 5.2 to addressing this problem. Section 5.3 then describes the heuristic cost function used to guide the search algorithm.

5.1 Dominant Paths

In order to determine the acoustic energy transfer from the source to the receiver, our proposed simulation model determines a subset of paths that the acoustic emergy may follow. These paths, termed *dominant paths*, are defined by two characteristic properties:

1. A dominant path is taken as an aggregate of all similar unique subpaths for the acoustic energy. Similar, in this case, means subpaths that share the same sequence

of edge deflections. For example, although there are infinitely many paths that a wave may take deflecting around a wedge, we use the single wave travelling the shortest path distance to represent all partial waves diffracting around the same sequence of wedges. We use the shortest path since an impulse (see Section 2.1.4) from the source will reach the receiver first by route of the shortest path. Because of this definition, no two dominant paths may share the exact same sequence of edge deflections.

2. Dominant paths are the most significant paths that meet the first criterion. To be more specific, we define significance here by the paths with the least loss-of-strength, and hence the greatest outcome in the signal.

Note that our use of the term dominant paths in this work differs from the use employed by Wölfle and Landstorfer [53] (Section 3.2).

In contrast to purely stochastic approaches, our approach based on dominant paths has a number of characteristics that are convenient:

- Dominant paths allow us to guide the stochastic sampling of the scene in the computation of the IR. As dominant paths represent an aggregate of many subpaths, by selecting only the most significant dominant paths, we have essentially also selected the most significant set of subpaths. As such, dominant paths allow us to completely skip sampling of areas with less utility, increasing the efficiency of the simulation.

- Dominant paths are easier to spatially understand and visualize. They represent significant flows of acoustic energy and they can be easily rendered to convey to modellers and artists what portions of the environment contribute to these acoustic flows.

- Each dominant path has a significant effect, and window of effect, on the resultant IR. By analyzing the final IR for the peaks of the contributions of each individual dominant path, one may draw a relationship between the parts of the final result and the dominant path that is responsible for the contribution.

For the latter two reasons, the dominant path model, in addition to functioning as an effective computational tool to simulating diffraction, provides a tool that modellers and artists alike may use for examining and controlling the results of the simulation for creative altercation.

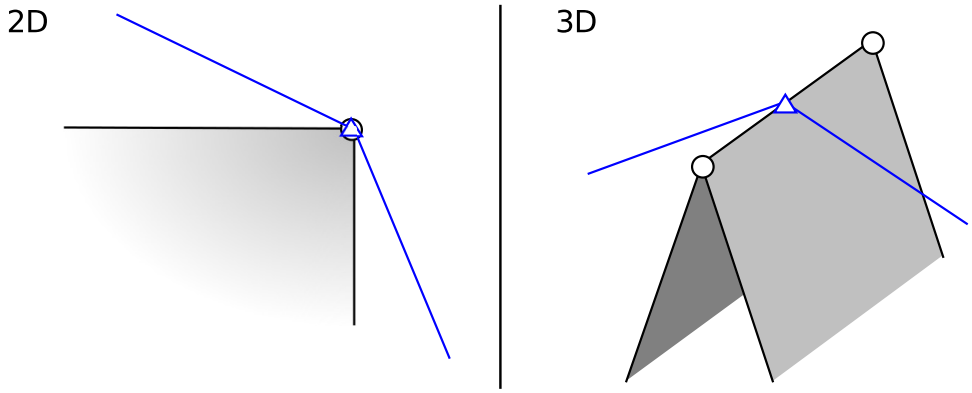

Figure 5.1: In 2D (left), a bend in the shortest path (indicated by the triangle) always corresponds to a vertex in the scene (indicated by the circle). In 3D (right), a bend may occur at an arbitrary point on a wedge.

5.2 Shortest Path Search

As we described previously, dominant paths are defined to be the most significant acoustic energy propagation paths that travel over a unique set of edges in the environment. The means by which we determine these paths is through the shortest path search algorithm.

The shortest path search is a common problem in artificial intelligence applications. In 2D, we can find the shortest path by constructing a path from the source to receiver through vertices in the scene graph. Since the number of vertices in the graph is finite, this path can easily be found in polynomial time, and $O(n \log n)$ solutions are possible [7].

In 3D, however, we observe that if we apply the same algorithm, the result at the end may not necessarily be the shortest path. This is because the shortest paths in 3D may deflect at locations other than vertices. More specifically, the shortest path may follow an edge (see Figure 5.1).

This result makes shortest path search a considerably harder problem in 3D. In fact, it has been proven that the shortest path solution is NP-Hard in 3D [7]. For this reason, we employ the technique demonstrated by Papadimitriou [34], which allows us to find an approximate shortest path solution in polynomial time.

We observe that the error between the ideal shortest path and the shortest path result of the vertex-driven 2D algorithm used in 3D is related to the maximum length of an edge segment. For this reason, Papadimitriou [34] prescribes an edge subdivision scheme that places vertices in the middle of an edge, allowing the classical algorithm to find solutions

closer to the ideal solution. The lengths of the subdivided segments are proportional to the distance between each edge segment and the origin. This is expressed as

$$l(\sigma) \leq \epsilon_1 D(s_0, \sigma), \tag{5.1}$$

where $l(\sigma)$ is the length of the edge segment denoted by σ, ϵ_1 is a constant, and $D(s_0, \sigma)$ is the distance between the origin (s_0) and the edge segment. This approach bounds the error of the solution to a factor of $(1 + \epsilon)$, where ϵ is chosen to adjust the desired error in the solution [7].

For acoustic simulation, the necessary precision to compute diffraction requires the error to be sufficiently smaller than the wavelength of the acoustic wave. Hence, the error must be bounded by λ/C, where C is a constant. Accordingly, given that the actual shortest length is l, the approximate path must have a length of $l_1 \leq l + \lambda/C$. Taking these factors into consideration, we have

$$l(1 + \epsilon) \leq l + \lambda/C \tag{5.2}$$

or

$$\epsilon \leq \lambda/(C \cdot l). \tag{5.3}$$

In order to minimize the error in all cases, we must substitute the maximum expected path length l_m, i.e., the error is bound by

$$\epsilon \leq \lambda/(C \cdot l_m). \tag{5.4}$$

We can see that, to maintain an accurate solution, we must reduce the error threshold, ϵ, as the maximum path length increases or as the wavelength decreases. It follows that there is a fundamental trade-off in the search algorithm for a given ϵ: the algorithm provides an accurate solution for a short distance at high frequency, or an accurate solution for a long distance at low frequency. As our method depends on this solution to the shortest path, the final solution given by our method is also bound by this correlation.

It is worth nothing, however, that as high-frequency waves diffract less than low-frequency ones, high-frequency signals will be lost over long distances when propagated through complex geometries with many deflections. Hence, the inaccuracy of high-frequency signals over long distances may be tolerated.

The algorithm developed by Papadimitriou [34] and applied by Choi et al. [7] recommends using breadth-first search (BFS) to find the shortest path. We make use of this recommendation. However, we do make modifications to the algorithm to allow for the finding of multiple dominant paths and the elimination of non-dominant paths. These changes are described in Section 6.2.

5.3 Diffraction Cost Function

Normally, the shortest path algorithm would employ the Euclidean distance heuristic as the cost function for performing the path search [4]. In determining the dominant paths, however, we must find the propagation paths that offer the most significant impact in our final result. Hence, we experimented with two different methods: a geometrical theory of diffraction (GTD) based signal strength estimate (described in Section 5.3.1) and the simplistic distance cost (described in Section 5.3.2).

Ultimately, we found that, while the GTD-driven estimate offered a theoretically more accurate selection of dominant paths, in practice, the results given by the distance metric performed nearly as well as the GTD-driven results, while computing in significantly less time.

5.3.1 GTD-Driven Signal Strength

As the GTD method [17] offers a faster method for computing diffraction, we considered the possibility of deriving an estimate of a path's strength.

To find the path of least signal loss, we must include the signal loss as a result of diffraction around edges as well as the loss due to distance. The GTD predicts that the diffraction of a ray hitting an edge forms a cone. The final amplitude following a diffraction event is then given by

$$u_e = \frac{D u_i e^{ikr}}{\sqrt{r}}, \tag{5.5}$$

where u_i is the complex wave state before deflection, k is a propagation constant given by $k = 2\pi/\lambda$, r is the propagation distance after deflection, and D is the diffraction coefficient [17]. For a wedge of angle $\theta_w = (2 - w)\pi$, the diffraction coefficient is given by

$$D = \frac{e^{i\pi/4}\sin(\pi/w)}{w\sqrt{2\pi k}\sin\phi}\left[\left(\cos\frac{\pi}{w} - \cos\frac{\gamma-\alpha}{w}\right)^{-1} \mp \left(\cos\frac{\pi}{w} - \cos\frac{\gamma+\alpha+\pi}{w}\right)^{-1}\right], \tag{5.6}$$

where $\alpha = \theta_S - \pi/2$, $\gamma = \theta_R - \pi/2$. The parameter ϕ is described in Figure 4.1, and the parameters θ_S and θ_R are described in Figure 4.2. Since the term e^{xi} is used to give the complex state containing the phase of the wave, we can compute the amplitude by simply removing this term. Hence, we have the amplitude coefficient for each node expressed as

$$A = \frac{\sin(\pi/w)}{w\sqrt{2r\pi k}\sin\phi}\left[\left(\cos\frac{\pi}{w} - \cos\frac{\gamma-\alpha}{w}\right)^{-1} \mp \left(\cos\frac{\pi}{w} - \cos\frac{\gamma+\alpha+\pi}{w}\right)^{-1}\right]. \tag{5.7}$$

Using this heuristic, our path search algorithm would find the path which maximizes the product of the amplitudes between the source S and any given node.

5.3.2 Distance

Alternatively, we also considered simply using the standard distance metric as a heuristic for the path search. Recall that the attenuation incurred on a signal is a function of the loss due to distance and the loss due to the diffraction events. Hence, the distance metric would properly account for the loss due to travel. Furthermore, since distance can be computed in a straight forward manner, this heuristic offers considerably faster computation.

5.4 Numerical Calculation of Impulse Response

In order to simulate the impulse response function for the diffraction event, $h(t)$, we apply our Monte-Carlo extension to the BTM method as described in Section 4.5. We remark that the continuous-time IR is given as an M dimensional integral over all possible combinations of edge-positions z_m. The derivative of this integral is given in Equation 4.29.

Chapter 6

Implementation

Our algorithm for computing the IR of an environment using dominant paths is divided into a three stage pipeline. The three stages are the preprocessing stage (Section 6.1), the path search stage (Section 6.2), and the computation stage (Section 6.3). This division is created to simplify the computation, to optimize performance (through caching) when sampling the environment multiple times, and to allow for programatic hooks for manipulation of the intermediate results. The programatic hooks are of particular significance as they enable one to analyze and manipulate the dominant paths for artistic experimentation.

It is worth noting that, in our implementation, the dominant path search algorithm is designed to originate the search at the source, working towards the receiver. However, it would be equally possible (and in some applications potentially advantageous) to originate the search at the receiver, working towards the source. Since the problem is symmetrical, such an implementation would be achieved by simply swapping the words source and receiver below.

6.1 Preprocessing Stage

The preprocessing stage is the first stage in the simulation pipeline. This stage reads in the 3D geometric scene representation and produces the search graph that will be used in performing the heuristic path search. This graph, which we refer to as the *edge visibility graph*, is a connected graph where nodes represent the centres of subdivided edges in the original geometric scene description.

The edge visibility graph itself is generated in two sub-stages: the reduction of the scene into an *edge cloud* (Section 6.1.1) and the extension of the edge cloud into the full edge visibility graph (Section 6.1.2). The process is summarized in Figure 6.1.

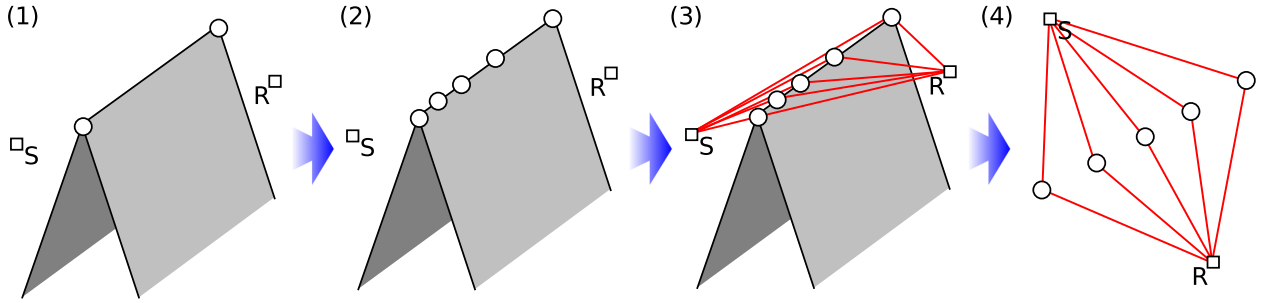

Figure 6.1: A diagram illustrating the steps by which the preprocessing stage converts a 3D scene into a search graph to be used by the path search stage for dominant path determination. Note that the source and receiver are represented by S and R, respectively. Initially, edges on the scene (1) are subdivided using the algorithm proposed by Papadimitriou [34] (2). The visibility of each edge node to every other node is then determined (3). Finally, the resulting data is used to construct the edge visibility graph (4).

6.1.1 Producing the Edge Cloud

The edge cloud, in effect, is a practical implementation of the edge-subdivision algorithm by Papadimitriou described in Section 5.2. The structure consists simply of a set of points in space. More specifically, these points represent the centre of the segments of subdivided edges in the scene. This subdivision is performed in line with the recommendations by Papadimitriou [34]. The origin of the edge, for our purposes, is chosen to be the point on the edge which is closest to the source, S, in the diffraction simulation. The edge is then divided by placing a number of points on the edge such that

$$p_i = \begin{cases} 0 & : i = 0 \\ \epsilon_1 D(s_0, \sigma)(1+\epsilon_1)^{i-1} & : i > 0 \\ -p_{-i} & : i < 0 \end{cases} \tag{6.1}$$

This process terminates when we reach the end of the edge. These points serve as possible locations through which a path (a candidate dominant path, in this case) may

travel. A caveat of this stage is that the two normals of the edge (the normals of each of the two faces bounded by the edge) must be preserved for processing in later stages. For this reason, we maintain an association between each point in the edge cloud and the original scene data-structure.

6.1.2 Producing the Edge Visibility Graph

Once the subdivision and generation of the edge cloud is complete, the edge visibility graph is constructed from the edge cloud. To perform this conversion, we perform an $n \times n$ visibility test between all points in the edge cloud against the original scene. We define these visibility relationships as *visibility segments*.

The computation of a full n to n visibility graph was chosen due to the simplicity of the algorithm for the demonstration of this approach. This leads to a worst-case processing and data-size for the visibility graph of $O(n^2)$. However, we note that the n to n connectivity matrix may be sparse with far fewer connections or may show characteristics of grouping. This is due to the fact that, in many environments, most vertices will be visible only by a subset of vertices in the scene. This is particularly noticeable in environments that have large obstacles or environments with several chambers. Examples of such environments include building interiors.

This observation means that we can optimize data-size by using structures that take advantage of the sparsity of the matrix by recording only the connected vertices. Similarly, computation could be optimized through approaches such as scene subdivision [27].

Since this phase depends only on the scene composition and the position of the source, it is not necessary to recompute the graph if only the receiver is moved.

6.2 Path search stage

The path search stage is the second main stage of computation, and it is responsible for taking the intermediate edge visibility graph data-structure and determining the dominant paths of sound between the source and receiver. At a high level, this stage consists of a modified heuristic BFS search algorithm. The heuristics employed in determining the dominant paths are described previously in Section 5.3.

In a traditional path search problem, it suffices to find one path between an origin and a destination. However, for the purposes of determining the multipath contribution

to the IR, we must be able to construct multiple dominant paths quickly and efficiently. While a candidate path may not be the globally strongest path, it may still constitute the locally strongest path across a unique sequence of wedges and may be strong enough to significantly affect the final IR. Such a path should still be selected as a dominant path. Hence, we employ a variation of the BFS algorithm. Our variation includes two important modifications:

1. To allow of the finding of multiple paths, it does not terminate after a single path is found. Instead, the termination is determined based on whether the corresponding results meet a minimal path strength threshold.
2. Upon the production of a path, the path must be rejected if it crosses the exact same sequence of edges as a previously produced path. Recall that a dominant path represents all subpaths that follow a unique sequence of edges. Hence, two paths that follow the same sequence of edges are part of the same dominant path.

The algorithm can be summarized with the following pseudocode:

```
DoPathSearch:
  paths <- Array();
  while (weakest path in paths).strength > threshold
    new_path <- edge_vis_graph.bfs_find_next();

    if( new_path.edges matches any path.edges in paths ):
        continue

    paths.append( new_path )
```

6.2.1 Search Termination Condition

As the strength of the resultant IR can vary greatly based on the problem characteristics, we choose the cutoff threshold as a fraction of the strength of the strongest dominant path. In this way, we neither select too many paths in a problem with a strong result, nor fail to select any paths in a problem in which all paths are relatively weak.

6.2.2 Path Rejection Algorithm

Since it would be computationally expensive to check many possible path results against the existing dominant path results, we make use of a tree data structure to perform path rejection.

As paths are constructed during the BFS algorithm, they are added to a *candidate path tree*. The nodes of the candidate path tree correspond to edge nodes in the edge visibility graph. The root of the tree is always the source, and the leaf nodes are always the receiver. A complete candidate dominant path, therefore, is represented as a path from the root of the tree to one of the leaves in the tree. Note that a single edge may appear in multiple nodes of the tree, as long each repetition of a node has a unique sequence of parent nodes.

This tree, therefore, allows us to quickly determine whether a particular sequence of edges from the edge has already been recorded with the following method:

```
RegisterOrReject:
  path_node <- candidate_path.source
  tree_node <- path_tree.root

  do:
    if( path_node.is_receiver? and tree_node.is_receiver? ):
      RejectPath()
    else if( tree_node.has_child( path_node.next.get_edge() ) ):
      path_node <- path_node.next
      tree_node <- tree_node.get_child( path_node.get_edge() )
    else:
      while not path_node.is_receiver?
        path_node <- path_node.next
        tree_node <- tree_node.add_child( path_node.next.get_edge() )
      AcceptPath()
```

6.3 Computation stage

The final stage in simulation is to compute the IRs of each dominant path and accumulate the results together to find the composite IR for the environment. At this stage, the FR may be optionally computed as well.

For each dominant path, the estimation of the IR is achieved through our Monte Carlo algorithm for performing higher order diffraction as described in Section 4.5.2. As the

algorithm depends on integrating over the length of the edges, we must resolve the edge nodes in the dominant paths back to the original edges of the environment geometry.

Before performing the Monte Carlo integration, however, we also precompute certain integration parameters that are relatively constant. By relatively constant, we mean that the value of the parameter is not a function of z, the sampling position on each edge. For example, θ_w and edge normals used to find angles remain constant throughout the Monte-Carlo sampling. Additionally, if two sequential edges are parallel in space, a few additional parameters, such as the diffraction angles and radii between the two edges, remain constant. Although, such precomputations are not necessary to the final result of the simulation, they can considerably improve performance.

One further adjustment that must be performed in this stage is to compensate for along-the-surface propagation. If the path travels against a surface, then Equation 4.29 must be multiplied by a factor of 2 for each segment of the dominant path which propagates against a surface. This is required due to a doubling of acoustic pressure when a source is mounted on a baffle [43].

Once the integration for each dominant path is complete, the final IR is thence obtained by accumulating each individual dominant path IR. The FR would finally be computed from the IR by means of a Discrete Fourier Transform.

Chapter 7

Results

In order to test the effectiveness of our simulator, we constructed several base environments to match different experiments conducted in the literature. Our simulator was then configured to run the simulations considering parameter values consistent with information provided in the literature. The IRs and FRs generated by our simulator were then compared qualitatively with the results provided in the reference experiments.

The first two experiments we conducted, covered in Section 7.1 and Section 7.2, are designed to test the accuracy of our Monte Carlo integration algorithm. Accordingly, we repeat two most basic test cases for the BTM approach against the original BTM integration. We then move on to the computation of composite IRs and dominant path searching by repeating the Medwin's Maekawa finite screen experiment in Section 7.3. Finally, we perform an investigation into determining the computational time and the quantitative accuracy of our Monte-Carlo sampling algorithm in Section 7.4 and conclude with an investigation of the multi-threaded performance of the same algorithm in Section 7.5.

7.1 Single Diffraction

The most basic diffraction case is diffraction across infinite and finite wedges as described in Section 2.1.4. In order to test the basic functionality of our simulator, we simulated a 90° Jonasson wedge, as described by Medwin et al. [24], between our listener and our receiver. The experimental set-up is given in Figure 7.1.

We present our results in Figure 7.2. Qualitatively, our results show the main features of the IR described in the literature [43]. Namely, the signal peaks at the time of the

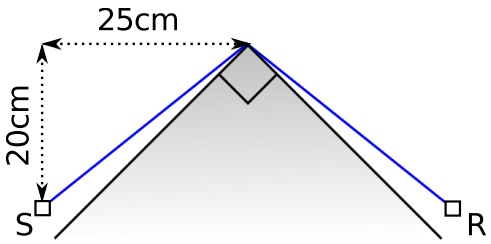

Figure 7.1: 90° Jonasson wedge used for impulse response simulations. The source (S) and receiver (R) are both placed on opposite sides of the wedge with a 25 cm horizontal displacement and 20 cm vertical displacement from the apex.

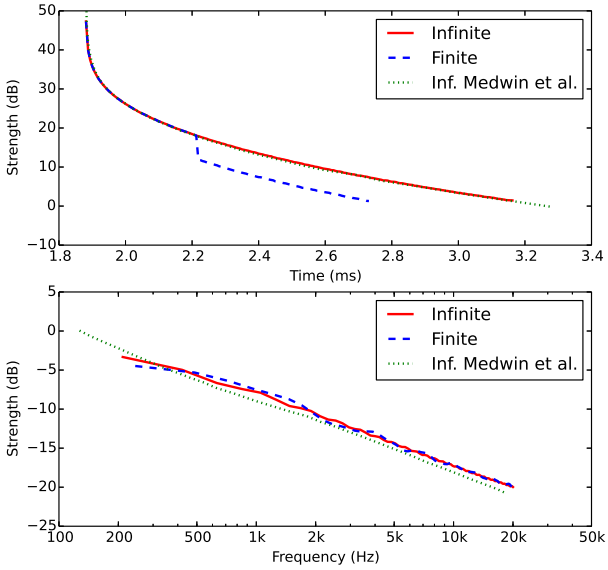

Figure 7.2: Simulation of the impulse response (top) and frequency response (bottom) for an infinite (red, solid) and finite (blue, dashed) Jonasson 90° wedge. The finite wedge mid-function partial loss of strength and terminal loss of strength are clearly visible at around 2.2 ms and 2.7 ms respectively. The infinite wedge simulation shows good agreement with a pure BTM implementation reported by Medwin et al. [24] (green, dotted).

minimal path, and it follows an exponential decline thereafter. In the finite wedge case, we further see the partial and full instantaneous losses of power at around 2.2 ms and 2.7 ms respectively. Our Monte Carlo results also show good agreement with results reported by Medwin et al. [24], calculated with a pure BTM implementation (refer to Section 4.2).

7.2 Double Diffraction

Multiple diffraction is a considerably harder problem than single diffraction due to the higher dimensions of integration. One of the main advantages of using a Monte Carlo based algorithm, as we described in Section 4.5, is that it allows the decoupling of the sampling strategy from the geometry of the problem. In this experiment, we verify that our integration algorithm accurately models second-order diffraction by repeating the experiments of double diffraction conducted by Medwin et al. [25]. The experimental set-up is given in Figure 7.3.

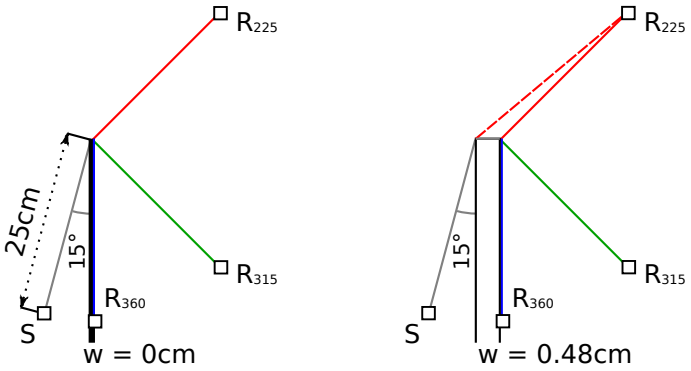

Figure 7.3: Experimental set-up of the double diffraction on a fixed wall scenario used by Medwin et al. [25]. The source (S) and receivers (R_{225}, R_{315}, and R_{360}) are placed 25 cm away from the nearest apex. The source and three receivers are placed at 15°, 225°, 315°, and 360° from the surface nearest to the source, respectively. The experiment is repeated for the single diffraction case for an infinitely thin wall (left), and for the double diffraction case with a wall 0.48 cm thick (right). Note that in the double diffraction 225° case, both a single (red, dashed) and a double (red, solid) diffraction path exist.

We include both the original simulation results by Medwin et al., which were produced using a modified BTM implementation (refer to Section 4.2.2 and Figure 4.4), and empirically measured samples also presented in the same paper. The results are displayed in

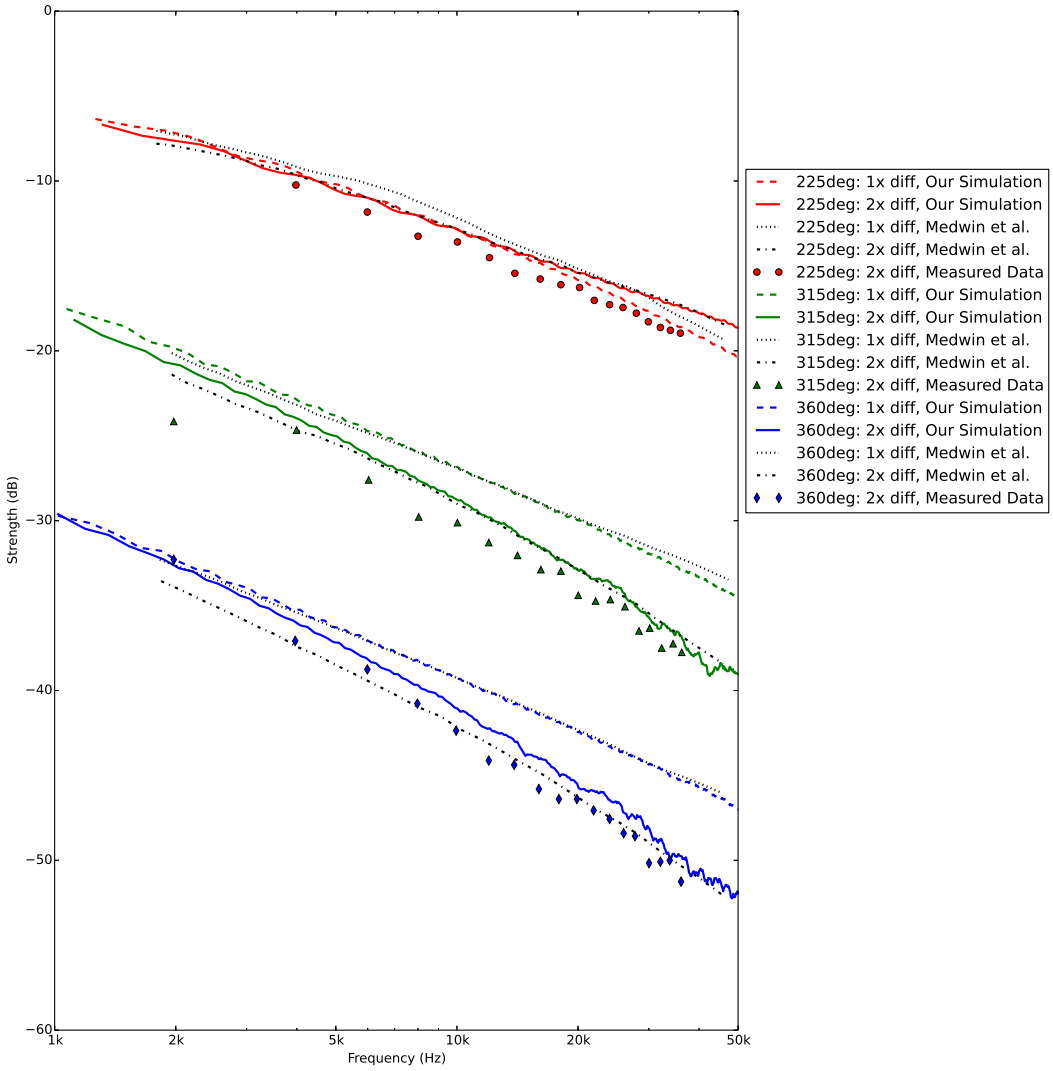

Figure 7.4: Simulation of double diffraction on a fixed wall as described by Medwin et al. [25]. Results for similar receiver positions are grouped together with results for the 225° case appearing at the top, the 315° case in the centre, and the 360° case at the bottom. Our results are depicted with red, green, and blue for the three cases respectively in both single (dashed) and double (solid) diffraction experiments. Results by Medwin et al. are depicted with black in both single (dotted) and double (dash-dotted) experiments. The circle, triangle, and diamond points depict measured data for the double diffraction case at the three receiver positions, respectively, as provided by Medwin et al. [25].

Figure 7.4. Note the close similarity between our Monte Carlo results and the BTM results for both single and double diffraction.

For double diffraction we note a slight discrepancy between our results and the results by Medwin et al. [25] at low frequencies, particularly in the 360° experiment. We, however, point out that when comparing the results against the measured data, our results show better agreement with the measured data. We make this assertion on the following observation. In the case of the 225° and the 315° experiments, when comparing the raw data to both our results and the results by Medwin et al., the simulation results tend to be slightly stronger than the measured results. We suspect this minute difference in strength is due to losses during measurement. In the case of the 360° experiment, the results by Medwin et al. show the opposite trend in the low frequencies: the simulated results are slightly weaker than the measured results. Our results, however, maintain the same trend as observed in the 225° and 315° experiments.

7.3 Maekawa Finite Screen

Our model is flexible and capable of simulating multiple dominant paths to return a composite IR. Furthermore, our staged dominant path determination algorithm and inter-stage hooks for further processing and alteration allow our design to be extended. To demonstrate these abilities, we reproduced the Maekawa finite screen scenario employed by Medwin et al. [24]. In the Maekawa Finite screen scenario, a thin screen is placed perpendicularly to a (relatively) infinite wall. The source is then placed above the screen, and two receiver positions are measured below the screen. The experimental set-up is given in Figure 7.5.

One notable requirement of this test is the simulation of reflection. Reflection is not directly implemented by our model. However, through the use of processing hooks, we were able to implement the image source method [1] in a python programming script to combine with our simulator and achieve the simulation of reflection/refraction paths.

In simulating reflection through the image source method, both the source and the receiver are mirrored on the opposite side of the reflecting wall. While the source and receiver's x and z positions remain unchanged, their y positions are reflected such that $y' = -y$. For each receiver position, the experiment is repeated four times:

1. from the source to the receiver,
2. from the mirrored source to the receiver,

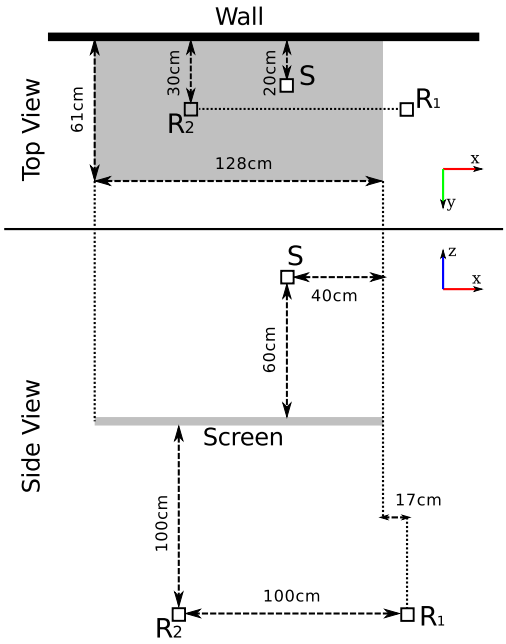

Figure 7.5: Layout of the Maekawa finite screen experiment described by Medwin et al. [24] as seen from above (top) and from the side (bottom). Source (S) and two Receiver (R_1 and R_2) positions are depicted.

3. from the source to the mirrored receiver, and

4. from the mirrored source to the mirrored receiver.

An orthographic view of the dominant paths computed during this simulation process is rendered in Figure 7.6 and Figure 7.7 for each of the two receiver positions respectively.

The computed IR and FR functions are plotted and compared against the results by Medwin et al. [24] in Figure 7.8. Our modeled results show good agreement with those provided by Medwin et al. for the IR. All peaks and qualitative features are captured in the simulation of both test conditions. Furthermore, the FR results show the same overall trends.

It is worth noting that our FR results are obtained through a Fourier transform of our IR results. The FR from Medwin et al., however, is obtained through a symbolic approximation derived from the pure BTM model. Furthermore, in the low frequencies, the results provided by Medwin et al. have limited resolution, which further contributes to the differences between both sets of results.

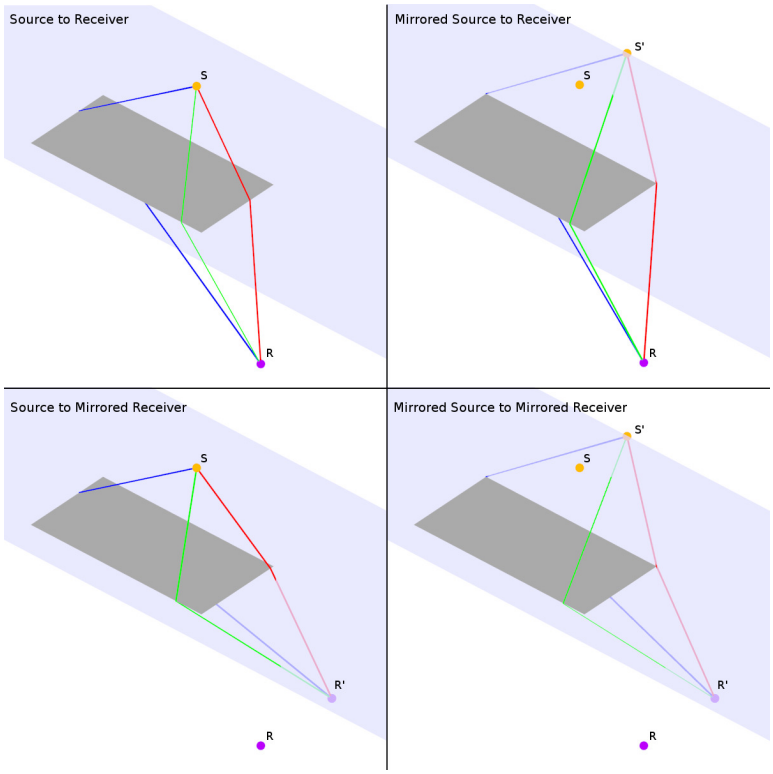

Figure 7.6: Orthographic rendering of the computed dominant paths (red, green, blue) for Maekawa finite screen scenario with the receiver in the first position. Simulation of reflection was achieved using the image source method [1]. Accordingly, four simulations were performed: source (S) to receiver (R), mirrored source (S') to receiver (R), source (S) to mirrored receiver (R'), and mirrored source (S') to mirrored receiver (R').

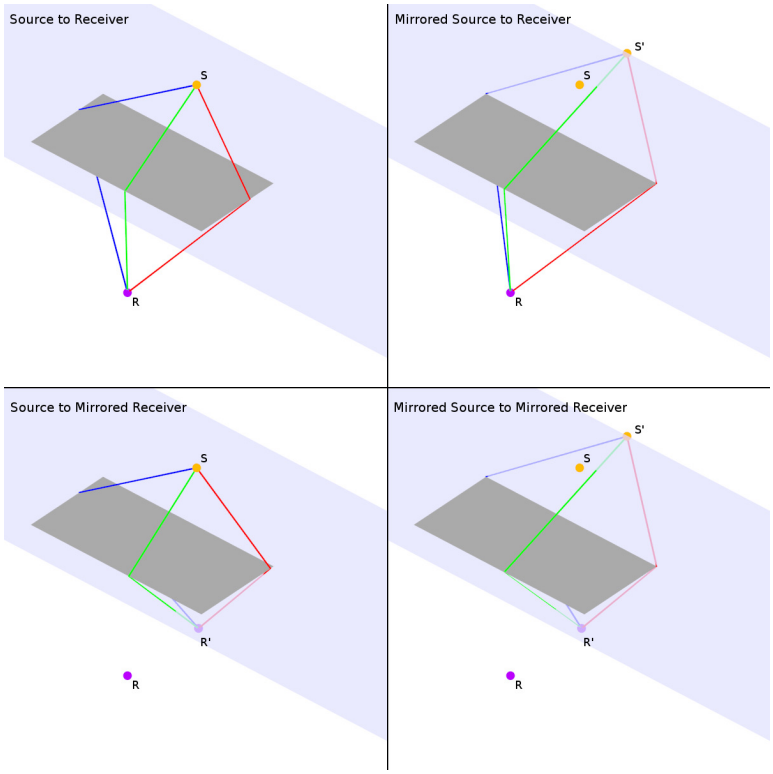

Figure 7.7: Orthographic rendering of the computed dominant paths (red, green, blue) for Maekawa finite screen scenario with the receiver in the second position. Simulation of reflection was achieved using the image source method [1]. Accordingly, four simulations were performed: source (S) to receiver (R), mirrored source (S') to receiver (R), source (S) to mirrored receiver (R'), and mirrored source (S') to mirrored receiver (R').

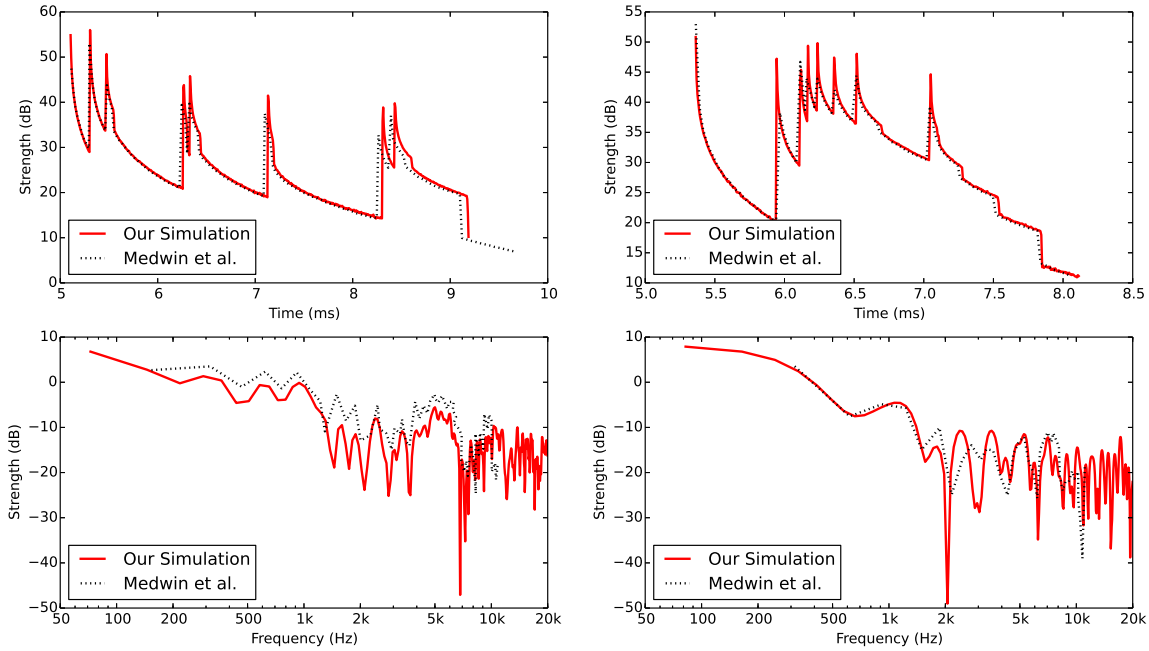

Figure 7.8: Comparison of our simulation results (red, solid) plotted against results provided by Medwin et al. [24] (black, dotted) for a Maekawa finite screen simulating two receiver positions. Both the impulse response results (top row) and frequency response results (bottom row) are depicted. In position 1 (left column), the receiver is placed under the screen, but near the edge. In position 2 (right column), the receiver is placed under the screen in a centered position.

Lastly, we demonstrate the ease of mapping the dominant path contributions to the final IR result by plotting the contributions of each dominant path against the IR in Figure 7.9. The colours used in plotting these contributions match the colours used in Figures 7.6 and 7.7 to emphasize the visual association between the computed dominant paths and their contributions.

7.4 Computation Time and Error

As our method makes use of Monte-Carlo random sampling, the error of the result is related to the number of samples used in the computation. Hence, there is a fundamental trade-off between computational time and computational error. Although the resulting

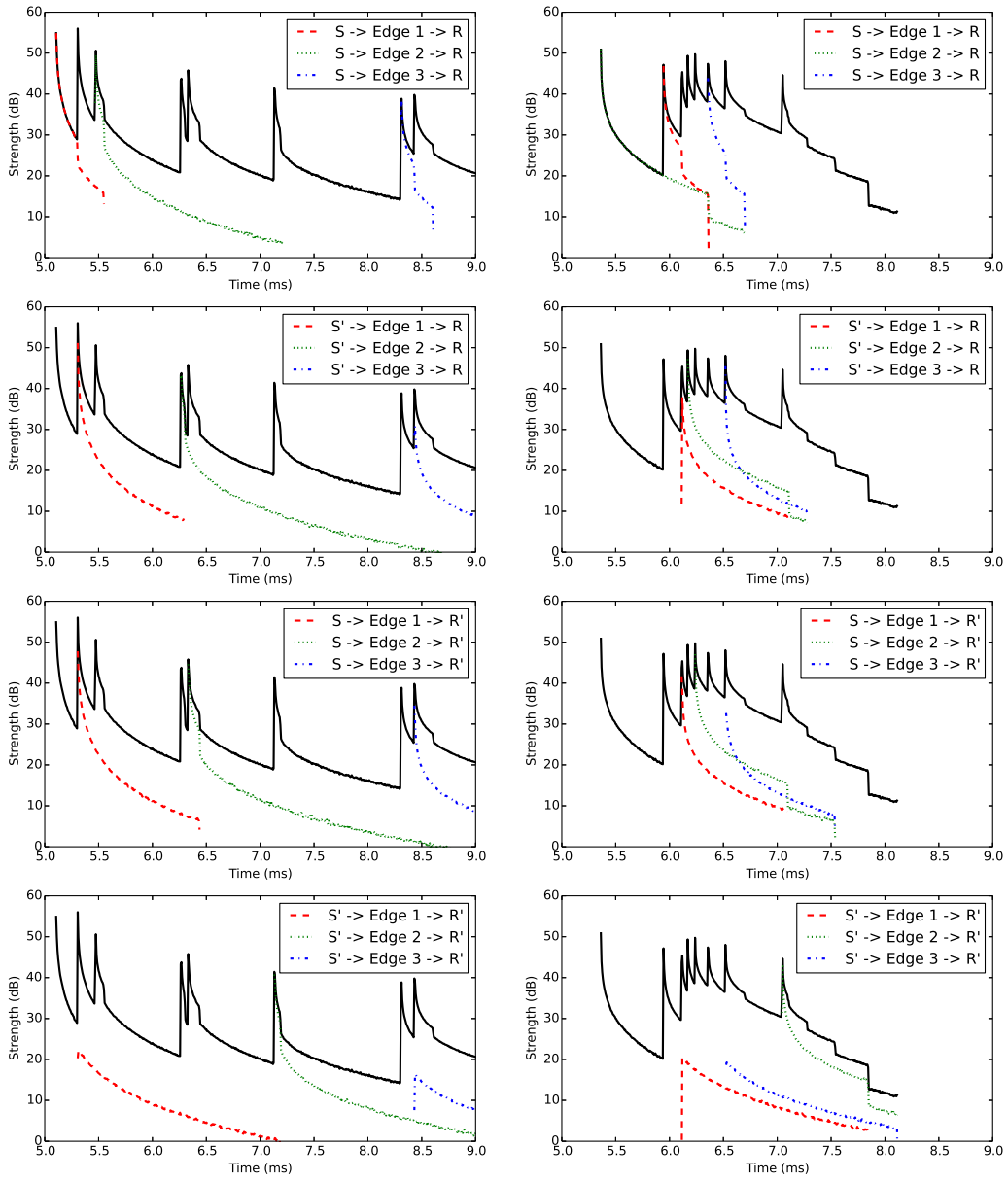

Figure 7.9: Demonstration of the contributions of each dominant path to the resultant impulse response. Position 1 is depicted on the left column, and Position 2 is depicted on the right column. Starting from the top we show the source to receiver paths, the mirrored source to receiver paths, the source to mirrored receiver paths, and finally the mirrored source to mirror receiver paths. The colours of the contributions were chosen to match the rendered dominant paths in Figures 7.6 and 7.7.

error of under-sampling is visible throughout the IR function, when we examine the FR, we find that the noise is more noticeable only at the higher frequencies. This observation is relevant since it allows us to attain accurate low frequency values with relatively few samples.

To demonstrate this observation, we re-ran the integration of the double diffraction experiment but varied the number of samples from the original 50 million samples down to 500 thousand samples. The results are depicted in Figure 7.10. As we can see, although the increased error due to under-sampling is clearly visible, it concentrates at the regions of higher frequencies.

We further ascertained this observation by computing the Mean Square-Error (MSE) in different frequency bands between the reference 50 million sample experiment and the under-sampled experiments. These results are tabulated in Table 7.1. As we can see, under-sampling by a factor of 100 leads in nearly no increase in error for the 0 – 1 kHz band. However, by the 2 – 5 kHz band, the MSE error is already greater than 1.0. On the other hand, the second set of results, which have been under-sampled by only a factor of 10, are able to keep the MSE below 1.0 up to a frequency of 20 kHz.

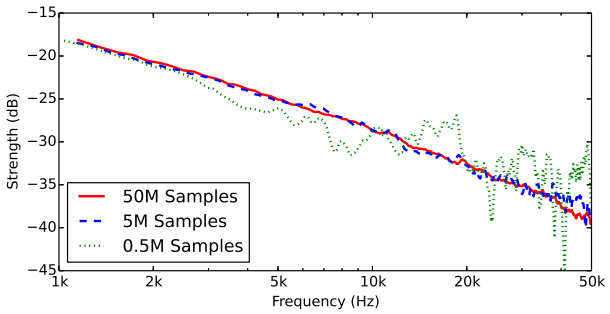

Figure 7.10: Comparison of the accuracy of the double-diffraction experiment by varying the number of samples from the original 50 million (red, solid), through 5 million (blue, dashed), down to 0.5 million samples (green, dotted).

The fact that we are able to obtain accurate results for the frequency response at lower frequencies using fewer samples has another practical implication. Recall that the UTD method makes a high-frequency, relatively infinite edge assumption (Section 3.3). While this assumption does not hold for lower frequencies, it does however hold well in high-frequency simulations. Hence, we find that our method is more accurate at computing low frequency results while the UTD method is more efficient at computing relatively accurate high-frequency results. This empirical observation suggests that the pairing of these

Samples	Time	MSE					
		1 kHz	2 kHz	5 kHz	10 kHz	20 kHz	50 kHz
50M	8.710 s	0	0	0	0	0	0
5M	0.9012 s	<0.001	0.007	0.015	0.174	0.226	2.163
500k	0.3144 s	0.395	0.721	2.317	5.170	9.636	23.404

Table 7.1: A comparison of the accuracy of the double diffraction experiment by varying the number of samples from the original 50 million down to 0.5 million. The relative error between the 50 million sample experiment and the 5 and 0.5 million sample experiments are shown as MSE measurements in different frequency bands starting with 0 – 1 kHz and ending with 20 – 50 kHz.

methods could lead to a fast and accurate computation of the entire frequency response.

7.5 Multi-threaded Performance

As modern processors have reached a ceiling in single processor computational power, it is advantageous to distribute simulations across multiple processing units. To test the suitability of our sampling method for multi-threaded computation, we modified our simulator to distribute the Monte Carlo sampling across multiple processing cores. We then tested our simulator across three hardware configurations:

1. A laptop computer containing an Intel Core 2 Duo processor at 1.86 GHz with two physical processing cores.
2. A consumer desktop computer containing an Intel Core i7 processor at 3.33 GHz with four physical processing cores and hyper-threading technology.
3. An enterprise-class workstation computer containing two Intel Xeon processors at 2.70 GHz with 12 physical processing cores each and hyper-threading technology.

The Core 2 processor is of a previous generation of processor design and we expect to see lower performance on this machine compared to the latter two, which both, despite the naming difference for enterprise marketing, share a similar architecture. Hyper-threading technology exposes two hardware threads per processing core to allow the processor to continue executing when a thread has halted while waiting for data from memory.

To further identify the gain this technique provides, we tested system 2 with hyper-threading both disabled (2a) and enabled (2b). Our raw results are given in Table 7.2. Note that the results show a nearly linear increase in time as the number of samples increase, as expected. This relation begins to break down at a lower number of samples because the initial path search phase of processing induces a constant cost regardless of the number of samples.

As each of the systems have different combinations of clock-speed and number of cores, we normalized the results to compare the systems on a per GHz per core basis in Table 7.3. We used the 50 million sample results to minimize the effect the constant path search cost in the result.

System	Configuration				Samples		
	Processor	Clock	Cores	HT	500k	5M	50M
1	Intel Core 2	1.86 GHz	2	No	1.475 s	10.09 s	100.3 s
2a	Intel Core i7	3.33 GHz	4	No	0.3545 s	1.317 s	12.99 s
2b	Intel Core i7	3.33 GHz	4	Yes	0.3144 s	0.9012 s	8.710 s
3	2x Intel Xeon	2.70 GHz	2x12	Yes	0.1871 s	1.172 s	1.789 s

Table 7.2: A comparison of the computational time required to run the simulation of the double diffraction experiment at differing number of samples on different hardware configurations. Note that the HT column depicts the availability or setting of hyper-threading technology on the machine.

System	Performance	Performance per Comp. Unit
1	498.5 kSamples/s	134.0 kSamples/s/GHz/Core
2a	3.849 MSamples/s	289.0 kSamples/s/GHz/Core
2b	5.741 MSamples/s	431.0 kSamples/s/GHz/Core
3	27.95 MSamples/s	431.3 kSamples/s/GHz/Core

Table 7.3: Comparison of the relative processing performance rates in computing the double diffraction experiment. The numbers are derived from running the 50M sample experiment. The results show significant performance gains in newer processor architectures and with the use of hyper-threading technology, as well as a linear increase in performance with increasing number of processing cores. The main features of each system are depicted in Table 7.2.

The results show significant performance gains from both architectural advancements between our laptop of a previous generation of processor design and the latter two systems.

The results also show a substantial gain in performance in using hyper-threading technology (nearly a 50% increase).

Between systems 2 and 3, with similar architectural design and processor configuration, we see a linear increase in performance despite the vastly different number of processing cores. This result suggests our Monte Carlo sampling method is highly suitable for parallel computation environments.

Chapter 8

Conclusions

In this work, we have developed an innovative solution for performing acoustic diffractive simulations within a ray tracing framework. Our work builds upon the Biot-Tolstoy-Medwin model of diffraction and accurate measurements of both the impulse and frequency responses of an environment throughout the spectrum – in both low-frequency simulations, and high-frequency simulations.

8.1 Contributions

Our contributions include a new Monte Carlo extension to the BTM model, a new *dominant path* approach to modelling the primary routes of acoustical energy, and a flexible simulator structure which can seamlessly be integrated into existing ray tracing frameworks and which allows for programmatic manipulation of the simulation process.

Our Monte Carlo sampling algorithm, used to integrate the diffraction line integral by Svensson et al. [43], is accurate to all acoustic wavelengths. Our algorithm is capable of working on arbitrary geometry over multiple arbitrary diffraction edges. Furthermore, the sampling strategy is decoupled from the geometry of the problem allowing one to easily chose the desired accuracy in the final result by varying the number of samples. The number of samples may even be chosen dynamically and simulation progress may be trivially monitored as all intermediate results constitute a candidate IR result.

Our sampling algorithm is also trivially parallelizable over a large number of processing units. We were able to distribute the integration process on 24 CPU cores simultaneously with nearly no loss in per-core performance.

Our dominant path method of modelling the transfer of acoustic energy gives our simulator several benefits. Dominant paths allow for a simple spatial representation of the acoustic energy transfer process. This spatial representation allows scientists and artists alike to analyze the routes that the acoustic energy takes through a scene. Furthermore, each dominant path can be trivially associated with the contributions the path makes to the final IR result of the simulation.

Our segmented processing pipeline allows for programmatic access to the intermediate results between stages. These programmatic hooks allow the model to be extensible and open the possibility to artistic manipulation. As demonstrated in Section 7.3, we used this capability to augment our simulation framework with the ability to simulate reflection with the image source method [1].

8.2 Limitations and Future Work

Our goal in this work was to develop and demonstrate a method to simulate diffraction for geometrical acoustics applications. Since there are many models of reflection, its inclusion in our dominant path formulation was not a priority. We assume that most geometric acoustics frameworks will already implement reflection. Hence, the dominant path framework focused on the simulation of paths involving diffraction. However, as paths with both diffraction and reflection can and do occur, it can be favourable to implement reflection within our dominant path framework. To this end, we briefly demonstrated how this method can be combined with the image source method for reflection in Section 7.3. In the future, however, we could explore a more integrative solution to the simulation of reflection.

APPENDICES

Appendix A

Documentation of Simulation Module

Our core simulation system is implemented in C++ as a module for the Python programming language. Invocation of each computational stage of simulation is thus started from a "driver" script. This appendix provides a brief introduction into the design and provides the API documentation for the core simulation module.

The core simulator is built upon a ray tracer originally intended for 3D graphics rendering. This ray tracer was initially designed as a project for the computer graphics course CS488 at the University of Waterloo. We decided to derive the acoustic simulation from a graphic ray tracer to demonstrate how our acoustic simulation model is designed in such a way to be compatible with ray tracers used in image synthesis applications.

The rest of this chapter contains the API documentation for the core simulation module. Section A.1 covers the portions of the API which remains largely unchanged from the original graphical ray tracing design. Section A.2 covers the new portions of the API which invoke the different computational stages.

Each sub-section of the following API documentation covers an object-oriented programming class in the API, with the exception of the last sub-section of each section. The last sub-section instead covers global functions which are not part of any class. Each sub-section begins with a brief description of the class it documents followed by the list of functions, with associated descriptions, in that class.

A.1 Modelling API

As the ray tracing module was originally designed for image synthesis, the environment, or *scene* in the ray tracer's terminology, is modelled using 3D primitives into a scene graph. The following interfaces are used for building, inspecting, and modifying the scene graph.

A.1.1 Class Vector3

The `Vector3` class represents a vector in 3D. It may be used to represent either a mathematical vector or a point.

Functions:

- **operator[]:** retrieves or sets the x, y, or z components of the vector.

A.1.2 Class SceneNode

The `SceneNode` class represents a node in the scene graph. Nodes are used to group a set of child nodes, geometry, and other specialized objects under a common set of geometrical transformations. Accordingly, nodes may be pure, they may contain geometry, or they may perform specialized functions.

Functions:

- **add_child(child):** method to add a child node to the current scene node.
- **scale(vector3):** performs a scale transformation on the current scene node and all its children.
- **translate(vector3):** performs a translation transformation on the current scene node and all its children.
- **rotate_x(angle):** rotates the current node and all its children around the x axis.
- **rotate_y(angle):** rotates the current node and all its children around the y axis.
- **rotate_z(angle):** rotates the current node and all its children around the z axis.

- **get_type():** gets the scene node's current type, where the type may be *NODE* (a pure scene node with no geometry), *GEOMETRY* (a node with geometry), *RENDER* (a special node used for graphical rendering, not used in acoustic simulation).

A.1.3 Class Scene

The `Scene` class manages the geometrical environment in which we will run our simulation. It stores the root node to the scene graph.

Functions:

- **cast_ray(v3_origin, v3_direction):** sends a single ray through the scene and returns the distance before the ray intersects with a surface in the scene.
- **render(render_node):** runs the main graphical rendering routine.
- **get_verts():** fetches all the vertices in the scene.
- **get_edges():** (new for acoustic simulation), fetches all the edges in the scene.
- **create_simulator(v3_source):** (new for acoustic simulation), creates and returns the main acoustic simulator object.

A.1.4 Global Functions

- **v3_compose(x, y, z):** creates a Vector3 object with the given coordinates.
- **make_node():** creates a standard scene node with no geometry.
- **make_geo_box():** creates a geometry node with a 2x2x2 unit cube geometry.
- **make_geo_plain():** creates a geometry node with a 2x2 unit plain geometry.
- **make_geo_obj(obj_path):** creates a geometry node with geometry loaded from an `.obj` file.

A.2 Acoustic Simulation API

The acoustic simulation API adds a set of interfaces to manage the acoustic simulation functions of the module. The following interfaces are provided to manage and manipulate the acoustic simulation.

A.2.1 Class Edge

The `Edge` class represents an edge in the scene.

Functions:

- **get_vertex_a():** returns one of the edge's two terminal vertices.
- **get_vertex_b():** returns the second of the edge's two terminal vertices.
- **get_vector_x():** returns a vector along one of the two faces joined by this edge.
- **get_vector_y():** returns a vector along the second of the two faces joined by this edge.
- **get_normal_x():** returns the normal of one of the two faces (the same face as returned by `get_vector_x()`) joined by this edge.
- **get_normal_y():** returns the normal of the second of the two faces (the same face as returned by `get_vector_y()`) joined by this edge.
- **get_normal():** returns a vector pointing in the direction the wedge is pointing.
- **get_edge_vector():** returns a vector in the direction of the edge.
- **get_wedge_angle():** returns θ_w.

A.2.2 Class Simulator

The `Simulator` class manages the core acoustic simulation functions.

Functions:

- **solve(v3_receiver):** executes the preprocessing stage and the path search stage of the simulation on the environment. Although these stages can be run independently, for our purposes we always ran these stages together and hence exposed only one call to run both stages.
- **free_space_attenuation():** once `solve()` has been called, this function returns the equivalent free space attenuation from the given source position to the given receiver position.
- **num_paths():** once `solve()` has been called, this function returns the number of dominant paths found in the environment.
- **print_paths():** once `solve()` has been called, this function nicely prints the dominant paths to `stdout`.
- **operator[]:** once `solve()` has been called, this function returns each dominant path.

A.2.3 Class Path

The `Path` class represents a *dominant path*. Dominant paths are retrieved from the simulator object after `solve()` is called to execute the preprocessing and path search stages.

Functions:

- **points():** returns a list of the dominant deflection points along the path.
- **value():** returns an estimate of the path's strength given by the GTD algorithm if the GTD heuristic is enabled (see Section 5.3.1).
- **length():** returns the path's total length.
- **get_max_time():** returns the maximum possible time of any subpath represented by this dominant path.
- **print_path():** prints a user-friendly representation of the path to `stdout`.
- **compure_ir():** performs the final stage of simulation on this dominant path to compute its contribution to the final IR.

- **compute_ir_medwin_1981():** computes the IR of this path using a pure implementation of the 1981 algorithm by Medwin et al. [24] (see Section 4.2). Only single-diffraction paths are supported.
- **compute_ir_sa():** computes the IR of this path using the sample-aligned algorithm (see Section 4.4.1). Only single-diffraction paths are supported.
- **compute_ir_even():** computes the IR of this path using the even-sized segment algorithm (see Section 4.4.2). Only single-diffraction paths are supported.
- **random_subpath():** computes a random sub-path represented by this dominant path and returns the set of points for the sub-path.

A.2.4 Class IrFunc

The `IrFunc` class represents the resultant IR function from integrating a dominant path. It is unitless: the values in the function are given as a ratio for an arbitrary input signal.

Functions:

- **clamp(threshold_ratio):** sets the minimum threshold for values in the IR function. Once the threshold is set, the tail of the function will be clipped off from the point the function's value falls below the threshold.
- **fft():** run a fast-Fourier transform on the function to produce the FR.
- **get_steps():** returns the number of steps / measurements stored in this object.
- **get_delta_t():** get the period of each step in the function. The period is the inverse of the sampling frequency.
- **get_max_time():** get the length of this function in time.

  `get_max_time()` = `get_delta_t()` × `get_steps()`.
- **operator[]:** get the sample at the given index.
- **operator():** get the sample at the given time.
- **operator+:** adds two `IrFunc` to produce the composite.

- **operator-:** subtracts an `IrFunc` from another.
- **operator*:** multiplies an `IrFunc` by a given constant.
- **operator/:** divides an `IrFunc` by a given constant.

A.2.5 Class FrFunc

The `FrFunc` class represents the resultant FR function produced by executing `fft` on an `IrFunc`.

Functions:

- **get_steps():** returns the number of steps / measurements stored in this object.
- **get_base_freq():** gets the width of each frequency band stored in this function. The frequency range stored at each index of this function is given by

  `get_base_freq()` × index – `get_base_freq()` × (index + 1).
- **set_free_space_attenuation(fsa):** sets the free space attenuation to anchor the results of this function to. Typically, frequency response results are given relative to the equivalent free space attenuation. Once this function is called with the appropriate value, the outputs of this object will be adjusted to compensate for the free space attenuation. A value of 1.0 means the given frequency band has the same strength as a wave which propagated through free space.
- **operator[]:** get the sample at the given index.

A.2.6 Class Environment

The `Environment` manages global simulation parameters.

Functions:

- **sound_v:** sets/gets c, the speed of sound, in m/s used in the simulation (default 340.29).

- **wave_lambda:** sets/gets the wavelength used for the GTD heuristic for the dominant path search (if the GTD heuristic is enabled).
- **step_t:** sets/gets the period of the samples (inverse of the sampling frequency) used for the IR results.
- **samples:** sets/gets the number of samples to use for IR integration.
- **threads:** sets/gets the number of processing threads to use for IR integration (by default set to the number of processing threads available on the machine).

A.2.7 Global Functions

- **get_env():** gets the `Environment` object used for configuring the acoustic simulation.

References

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

[2] G. E. Athanasiadou. Incorporating the Fresnel zone theory in ray tracing for propagation modelling of fixed wireless access channels. In *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, pages 1–5, sept. 2007.

[3] M. A. Biot and I. Tolstoy. Formulation of wave propagation in infinite media by normal coordinates with an application to diffraction. *The Journal of the Acoustical Society of America*, 29(3):381–391, 1957.

[4] R. L. Burden and J. D. Faires. *Numerical Analysis*. PWS-KENT Publishing Company, Boston, fifth edition, 1993.

[5] P. T. Calamia and U. P. Svensson. Fast time-domain edge-diffraction calculations for interactive acoustic simulations. *EURASIP J. Appl. Signal Process.*, 2007(1):186–186, January 2007.

[6] S. Chattopadhyay and A. Fujimoto. Bi-directional ray tracing. In TosiyasuL. Kunii, editor, *Computer Graphics*, pages 335–343. Springer Japan, 1987.

[7] J. Choi, J. Sellen, and C. K. Yap. Approximate Euclidean shortest path in 3-space. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 41–48. ACM, 1994.

[8] National Defence Research Comittee. *Physics of Sound in the Sea*. National Defence Research Comittee, 1946.

[9] F. Frassinetti, N. Bolognini, and E. Làdavas. Enhancement of visual perception by crossmodal visuo-auditory interaction. *Experimental Brain Research*, 147(3):332–343, 2002.

[10] E. R. Freniere, G. G. Gregory, and R. A. Hassler. Edge diffraction in Monte Carlo ray tracing. In *Proceedings of SPIE*, volume 3780, pages 151–157, 1999.

[11] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 21–32, New York, NY, USA, 1998. ACM.

[12] T. Funkhouser, N. Tsingos, I. Carlbom, G. Elko, M. Sondhi, J. E. West, G. Pingali, P. Min, and A. Ngan. A beam tracing method for interactive architectural acoustics. *The Journal of the Acoustical Society of America*, 115(2):739–756, 2004.

[13] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Methuen, 1964.

[14] K. H. Höhne, B. Pflesser, A. Pommert, M. Riemer, R. Schubert, T. Schiemann, U. Tiede, and U. Schumacher. A realistic model of the inner organs from the visible human data. In ScottL. Delp, AnthonyM. DiGoia, and Branislav Jaramaz, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2000*, volume 1935 of *Lecture Notes in Computer Science*, pages 776–785. Springer Berlin Heidelberg, 2000.

[15] D. L. James, J. Barbič, and D. K. Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics*, 25(3):987–995, July 2006.

[16] B. Kapralos, M. Jenkin, and E. Milios. Acoustical diffraction modeling utilizing the Huygens-Fresnel principle. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE 2005)*, Ottawa, Ontario, Oct 2005.

[17] J. B. Keller. Geometrical theory of diffraction. *The Journal of the Acoustical Society of America*, 52(2):116–130, Feb 1962.

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

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

[20] Andrzej Kulowski. Algorithmic representation of the ray tracing technique. *Applied Acoustics*, 18(6):449–469, 1985.

[21] C. Lauterbach, A. Chandak, and D. Manocha. Interactive sound rendering in complex and dynamic scenes using frustum tracing. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1672–1679, 2007.

[22] J. P. Lindsey. The Fresnel zone and its interpretive significance. *The Leading Edge*, 8(10):33–39, 1989.

[23] G. Mastoropoulou, K. Debattista, A. Chalmers, and T. Troscianko. The influence of sound effects on the perceived smoothness of rendered animations. In *Proceedings of the 2nd Symposium on Applied Perception in Graphics and Visualization*, APGV '05, pages 9–15, New York, NY, USA, 2005. ACM.

[24] H. Medwin. Shadowing by finite noise barriers. *The Journal of the Acoustical Society of America*, 69(4):1060–1064, 1981.

[25] H. Medwin, E. Childs, and G. M. Jebsen. Impulse studies of double diffraction: A discrete Huygens interpretation. *The Journal of the Acoustical Society of America*, 72(3):1005–1013, 1982.

[26] R. Mehra, N. Raghuvanshi, L. Antani, A. Chandak, S. Curtis, and D. Manocha. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Transactions on Graphics*, 32(2):19:1–19:13, April 2013.

[27] Michael Meißner, Dirk Bartz, Tobias Hüttner, Gordon Müller, and Jens Einighammer. *Generation of subdivision hierarchies for efficient occlusion culling of large polygonal models*. WSI, 1999.

[28] H. P. Moravec. 3D graphics and the wave theory. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '81, pages 289–296, New York, NY, USA, 1981. ACM.

[29] W. Moss, H. Yeh, J. M. Hong, M. C. Lin, and D. Manocha. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Transactions on Graphics*, 29(3):21:1–21:13, July 2010.

[30] S. Müller. Measuring transfer-functions and impulse responses. In D. Havelock, S. Kuwano, and M. Vorländer, editors, *Handbook of Signal Processing in Acoustics*, pages 65–85. Springer New York, 2008.

[31] S. T. Neely and J. B. Allen. Invertibility of a room impulse response. *The Journal of the Acoustical Society of America*, 66(1):165–169, 1979.

[32] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. In L. B. Wolff, S. A. Shafer, and G. Healey, editors, *Radiometry*, pages 94–145. Jones and Bartlett Publishers, Inc., USA, 1992.

[33] Brett Ninness. *Fundamentals of Signals, Systems and Filtering*. School of Electrical Engineering and Computer Science, The University of Newcastle, Australia, 2005.

[34] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20(5):259–263, 1985.

[35] 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, 2009.

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

[37] Z. Ren, H. Yeh, and M. C. Lin. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics*, 32(1):1:1–1:16, February 2013.

[38] S. W. Rienstra and A. Hirschberg. *An Introduction to Acoustics*. Eindhoven University of Technology, Mar 2014.

[39] A. Rimell and M. Hollier. The significance of cross-modal interaction in audio-visual quality perception. In *IEEE 3rd Workshop on Multimedia Signal Processing*, pages 509–514, 1999.

[40] N. Röber, U. Kaminski, and M. Masuch. Ray acoustics using computer graphics technology. In *10th International Conference on Digital Audio Effects (DAFx-07), S*, pages 117–124. Citeseer, 2007.

[41] A. Schmitz and L. Kobbelt. Wave propagation using the photon path map. In *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, PE-WASUN '06, pages 158–161, New York, NY, USA, 2006. ACM.

[42] A. Schmitz and M. Wenig. The effect of the radio wave propagation model in mobile ad hoc networks. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, MSWiM '06, pages 61–67, New York, NY, USA, 2006. ACM.

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

[44] M. T. Taylor, A. Chandak, L. Antani, and D. Manocha. RESound: interactive sound rendering for dynamic virtual environments. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 271–280, New York, NY, USA, 2009. ACM.

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

[46] F. Trebien and M. M. Oliveira. Realistic real-time sound re-synthesis and processing for interactive virtual worlds. *The Visual Computer*, 25(5-7):469–477, May 2009.

[47] N. Tsingos, I. Carlbom, G. Elbo, R. Kubli, and T. Funkhouser. Validating acoustical simulations in Bell labs box. *IEEE Computer Graphics and Applications*, 22:28–37, July 2002.

[48] N. Tsingos, C. Dachsbacher, S. Lefebvre, and M. Dellepiane. Instant sound scattering. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR '07, pages 111–120, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[49] N. Tsingos, T. Funkhouser, A. Ngan, and I. 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.

[50] N. Tsingos and J. D. Gascuel. Soundtracks for computer animation: Sound rendering in dynamic environments with occlusions. In *Graphics Interface '97*, Kelowna, Canada, 1997.

[51] V. Červený. Fresnel volume ray tracing. *Geophysics*, 57:902–915, Jul 1992.

[52] Jean Vroomen, Beatrice De Gelder, et al. Sound enhances visual perception: Cross-modal effects of auditory organization on vision. *Journal of Experimental Psychology Human Perception and Performance*, 26(5):1583–1590, 2000.

[53] G. Wölfle and F. M. Landstorfer. Field strength prediction with dominant paths and neural networks for indoor mobile communication. *MIOP '97, Mikrowellen und Optronik, 9. Kongremesse fr Hochfrequenztechnik, Funkkommunikation und elektromagnetische Vertrglichkeit, Sindelfingen, pp. 216-220*, 1997.

[54] C. Zheng and D. L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics*, 29(4):69:1–69:13, July 2010.

Index

Aliasing, 29

Camera, 11
Candidate path tree, 46

Diffraction, 8
Dominant path, 36

Edge cloud, 42
Edge subdivision, 39
Edge visibility graph, 42
Error, 38, 56
even-sized segments, 30

Frequency response, 9

Geometric Models, 15
Geometric Theory of Diffraction, 17

Huygen's approach, 25

Image source method, 52
Impulse response, 8
Intensity Level, 6

Line integral, 25

Monte-Carlo sampling, 31
Multi-threading, 59

Numerical Models, 15

Performance, 59
Power Level, 6

Ray tracing, 11
Real-time computation, 18
Room effects, 9

Sample-aligned segments, 29
Sound, 4
    Amplitude, 6
    Pressure, 6
    Speed, 5
Sound Pressure Level, 6
Sound Propagation, 15
Sound Synthesis, 14

Universal Theory of Diffraction, 17

Visibility segments, 44

fin