A Physically-Informed Prioritized Beam Tracer

Harish Venkatesan

Music Technology Area, Department of Music Research
Schulich School of Music
McGill University
Montreal, Canada

November 2019

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts

Abstract

The use of room acoustic modeling techniques in real-time interactive applications such as video games and virtual reality has been limited, mainly owing to their high computational costs. Among the various room acoustic modeling techniques, beam tracing is the fastest known deterministic geometrical acoustic technique for modeling specular reflections and has been in common use in various real-time room acoustic auralization engines. Yet, due to its moderately high computational costs, it imposes strict limits on the rate of movement of the sound source and change in the room geometry for seamless auralization of dynamic environments. In this thesis, we propose a physically-informed prioritized beam tracer that focuses on tracing the more relevant beams to reduce the overall computational cost further and hence ease some of these limitations.

The proposed algorithm judiciously traces beams by prioritizing them based on estimates of the energy that they carry regardless of the order of reflection. We use three factors to determine the priority of a beam, namely, the width of the beam, attenuation of energy due to transmission in atmosphere and loss of energy at the reflecting surfaces, which together determine the energy of a beam.

We present a performance comparison between the proposed beam tracer and some conventional approaches to beam tracing that have been previously published in literature. The beam tracers are evaluated in three different room models of varying levels of complexity and are compared based on two aspects: the number of beams required and the time taken to provide an acceptable level of accuracy of specular reflection energy. We also explore different conditions for termination of the beam tracing process, evaluating the pros and cons of each approach. Finally, we compare the mentioned priority factors on their influence on the beam tracer's performance.

The results show a greater computational gain for more complex rooms compared to simple, convex room models, requiring up to 10 times less beams compared to the conventional breadth-first and the pseudo-breadth-first approach to detect significant specular reflection paths. The beam tracers are also evaluated for use in real-time simulations with iterative refinement of the obtained solution and the timing results show that the prioritized beam tracer performs up to 19 times faster than the pseudo-breadth-first beam tracer.

Résumé

L'utilisation de techniques de modélisation acoustique des salles dans des applications interactives en temps réel telles que les jeux vidéo et la réalité virtuelle a été limitée, principalement en raison des coûts de calcul élevés. Parmi les différentes techniques de modélisation acoustique des salles, le traçage du faisceau est la technique acoustique géométrique déterministe la plus rapide connue pour la modélisation des réflexions spéculaires. Elle est couramment utilisée dans divers moteurs d'auralisation acoustique des salles en temps réel. Cependant, en raison de ses coûts de calcul modérément élevés, il impose des limites strictes au taux de déplacement de la source sonore ainsi qu'aux changements géométriques de la salle pour une auralisation harmonieuse des environnements dynamiques. Dans cette thèse, nous proposons un traceur de faisceaux hiérarchisé prioritaire basé sur des informations physiques qui se concentre sur le traçage des faisceaux les plus pertinents afin de réduire davantage le coût de calcul global et donc d'atténuer certaines de ces limitations.

L'algorithme proposé trace judicieusement les faisceaux en les hiérarchisant en fonction d'estimations de l'énergie qu'ils transportent, quel que soit leur ordre de réflexion. Nous utilisons trois facteurs pour déterminer la priorité d'un faisceau, à savoir la largeur du faisceau, l'atténuation de l'énergie due à la transmission dans l'atmosphère et la perte d'énergie sur les surfaces réfléchissantes, qui déterminent l'énergie d'un faisceau.

Nous présentons une comparaison de performances entre le traceur de faisceau proposé et certaines approches classiques du traçage de faisceau publiées antérieurement dans la littérature scientifique. Les traceurs de faisceaux sont évalués dans trois modèles de salle de niveaux de complexité différents et sont comparés en fonction de deux aspects: le nombre de faisceaux requis et le temps nécessaire pour fournir un niveau de précision acceptable de l'énergie de réflexion spéculaire. Nous explorons également différentes conditions pour mettre fin au processus de traçage du faisceau, en évaluant les avantages et les inconvénients de chaque approche. Enfin, nous comparons les facteurs de priorité mentionnés concernant leur influence sur les performances du traceur de faisceau.

Les résultats montrent un gain de calcul plus important pour des pièces plus complexes par rapport aux modèles de pièces simples et convexes, nécessitant jusqu'à 10 fois moins de faisceaux par rapport à l'approche conventionnelle de recherche de parcours en largeur et pseudo recherche de parcours en largeur pour détecter des trajets de réflexion spéculaires significatifs. Les traceurs de faisceaux sont également évalués pour une utilisation dans des simulations en temps réel avec un raffinement itératif de la solution obtenue. Les résultats de synchronisation montrent que le traceur de faisceaux hiérarchisé est jusqu'à 19 fois plus rapide que le traceur de faisceaux utilisant un pseudo algorithme de parcours en largeur.

Acknowledgments

I would like to thank my supervisor, Prof. Gary Scavone, for his continual support and guidance that led to the successful and timely completion of this thesis. I very deeply appreciate the freedom he gave to me in choosing the topic of my research exploration and his constant constructive feedback guiding me throughout the journey of completion of this thesis. I would like to extend my heartfelt gratitude to my co-supervisor, Dr. Esteban Maestre, for the many insightful conversations and discussions we had regarding the research and for helping me stay focused on the important aspects of this thesis, keeping me on the right track.

I would like to thank Darryl Cameron at Music Tech and Yves Méthot at CIRMMT for their support with absolutely any tech related issues I may have run into during my two years at McGill University. I would also like to thank my friends and colleagues at CAML and rest of Music Tech for all the wonderful conversations regarding culture, food, technology and many more topics that made me feel like a part of a very diverse bunch of people with a common passion towards music technology. It was really great getting to know you all. My journey would not be complete without all the friends I have made in Montréal from whom I learned a lot. They definitely deserve a mention.

Last, but not the least, no amount of words can truly describe my appreciation and gratitude for my parents, sister, grandmothers and girlfriend, who have been with me through the ups and downs in these past two years that I have been away from home. They have been the most instrumental in my success and have always motivated me to pursue my passion and perform to the fullest of my capabilities, celebrated even the tiniest of my accomplishments and have never failed to extend a loving and caring arm during the toughest of times when I have needed them the most. I shall forever remain indebted to you for your unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sounds produced in the real world reach our ears after being processed by the surrounding environment, i.e., the sounds we hear depend on our immediate surroundings. For example, in a concert hall, the sound of an orchestra reaches us after bouncing off the structures in the hall in complicated ways, undergoing transformations on each interaction with the surfaces. This causes the sustain and the gradual decay of the sounds that we hear, which is known as reverberation (commonly known as reverb) of sound. The importance of reverberation has been understood for ages and we can find evidence for this in ancient monuments and architectural wonders world-wide. In the more recent times, the advent of recording technology brought in tremendous interest in trying to emulate reverberation using electronic circuits and computers. In the realm of digital audio, since Schroeder's [2] first publication of the all-pass filter based artificial reverb algorithm in 1960 , there have been many advancements in digital reverbs, from various delay-line based implementations such as the ones presented by Moorer [3] and feedback delay networks by Jot and Chaigne [4], convolution reverbs (Reilly and McGrath [5]) that use recorded impulse responses of real rooms, to plate (Bilbao et al. [6]) and spring reverb simulations by physical modeling. However, these techniques fall short when it comes to truly physically coherent and realistic rendering of room acoustics considering source and listener dynamics (location and movement) and are mainly used as effects in audio production.

Room acoustic modeling aims to accurately model the acoustics of spaces using the knowledge of acoustic wave propagation and simulating the acoustic phenomena that take place in the real world, such as reflection, absorption, diffraction, etc. It has found applications in several fields including architectural acoustics, video games and virtual reality. Room acoustic modeling has aided in the field of architectural acoustics, helping architects design better rooms by estimating how they might sound even before actually constructing them. In interactive applications such as video games and virtual reality, it is essential for the visual and auditory representations of the space to tell the same story and not conflicting ones in order to make the experience engaging for the user. Room acoustic modeling aims at accurately rendering the acoustics of the virtual environment in order to achieve true immersion (the feeling of presence in the virtual world) of the user in the virtual space. The requirements of the two fields are very different. In architectural acoustics, it is very important to accurately model the acoustics of the designed space, while in interactive applications, it is essential to maximize the accuracy while minimizing the latency of auditory rendering. However, the use of room acoustic modeling techniques in real-time interactive applications has been severely limited due to the trade-off between computational cost and accuracy.

1.1 Motivation

There are two main categories of room acoustic modeling techniques: Geometrical Acoustics (GA) and Numerical Acoustics (NA). GA [7] are a class of room acoustic modeling techniques where rectilinear propagation of acoustic waves is assumed. Although GA techniques are less accurate compared to numerical methods, which involve solving the wave equation at sampled points in space, they are a popular choice for interactive applications as they are far less computationally expensive. Nevertheless, there have been attempts in the past at applying numerical methods for interactive applications [8] due to their high degree of accuracy. However, they work only in static rooms, i.e., they fail when rooms dynamically change during user interaction (such as opening of doors and portals, breaking of walls and barriers, etc).

Beam tracing is a GA technique that is known to model specular reflections (mirror-like reflections by smooth surfaces) in a deterministic manner at the least computational cost. The algorithm consists of two main steps: visibility map construction of the space from the perspective of the source through reflections (beam tree construction) and the identification of unobstructed reflection paths from the source to the listener (path validation). Among the two processes, the beam tree construction step is the most computationally demanding and the algorithm suffers from restrictions on the rate of movement of the sound source and the rate of change of room geometry for smooth acoustic rendering of the virtual environment.

There have been several works published in the past addressing the cost of the beam tree construction step [1, 9, 10, 11, 12]. However, they have been either applicable only to static rooms or add restrictions on the movement of the listener. The conventional beam tracing algorithm may be made more efficient by avoiding some unnecessary computations. In this thesis, we present a technique which avoids these unnecessary computations by judiciously tracing beams to identify reflection paths in fewer iterations.

1.2 Project Overview

The beam tracer presented in this thesis makes use of physically informed energy estimates to prioritize tracing of beams with higher energies and guide the growth of the beam tree in directions where there is a greater likelihood of detecting significant reflection paths between the source and the listener. We recognize three factors that affect the energy carried by a beam: the width of the beam, attenuation due to air absorption and energy loss at reflecting surfaces. At every step of the beam tracing process, the beam with the highest priority/energy is traced further in the hope of finding valid reflection paths earlier. The prioritized beam tracer is designed in such a way that its tolerance to listener movement as in the conventional beam tracer is retained and the tolerance to source movement is improved by reduction in computations.

We present a thorough evaluation of the physically informed prioritized beam tracer, comparing its performance to conventional approaches to beam tracing such as the depth-first and breadth-first beam tracing in room models of different complexities and occlusion conditions. We also compare the presented beam tracer with the pseudo-breadth-first approach (an adaption of the depth-first beam tracer to provide paths in a breadth-first manner) used in the EVERTims real-time beam tracing application [13, 14]. In this thesis, we also explore different termination conditions for stopping the process of prioritized beam tracing such as fixed number of beams, priority thresholds and hybrid techniques employing adaptive pausing conditions with fixed minimum priority thresholds.

The algorithms are compared based on the distribution of paths in the beam tree to verify that the prioritized beam tracer detects valid reflection paths earlier than the other algorithms, showing greater compactness of distribution towards the beginning of the tree. The main aim of this comparison is to verify that the proposed beam tracer requires fewer beams than the other algorithms. Simulation results show that the prioritized beam tracer requires between 1.75 - 9 times fewer beams compared to the breadth-first beam tracer and 4 - 11 times fewer beams compared to the pseudo-breadth-first beam tracer for detecting significant amount of specular reflection energy, depending on the complexity of the room geometry and the occlusion condition.

The beam tracers are also compared based on timing performance for a real-time context that uses iterative refinement of the specular reflection energy, like in the EVERTims software package. The simulation results show that the prioritized beam tracer performs up to 19 times faster than the other beam tracers in tracing significant amount of specular reflection energy. Finally, we present, in brief, a comparison on the extent to which different priority factors influence the results of the prioritized beam tracer. Results show that the beam-width factor is the most important, followed by the material-absorption factor and then the air-absorption factor in the room geometries that were considered in this thesis.

1.3 Thesis Overview

In Chapter 2, a brief overview of room acoustic modeling, mainly GA techniques is presented, followed by a detailed explanation of beam tracing, commenting on the advantages, shortcomings and challenges faced in the algorithm, including possible methods to circumvent some of these challenges.

In Chapter 3, the computation of prioritization factors is described in detail, along with the interpretation of the priorities. A discussion on the possible termination conditions for the prioritized beam tracer and their implications is also presented.

In Chapter 4, a thorough description of the evaluation platform and room geometries chosen for simulation with the proposed beam tracer is given, along with the analysis of the obtained simulation results and a comparison of its performance against other beam tracers. A brief report on the comparison of the prioritization factors is also presented at the end of this chapter.

Finally, Chapter 5 concludes by summarizing the results, observations and inferences drawn from the performance of the prioritized beam tracer compared to the other beam tracers and its suitability for real-time applications. Future directions of the project are also discussed in this chapter.

Chapter 2

Background

In this chapter, we give a brief overview of room acoustic modeling and some common geometrical acoustic techniques used for room simulation. We explain in detail the functioning of the beam tracing algorithm and the challenges faced.

2.1 Room Acoustic Modeling

Acoustic waves emitted by a sound source interact with the objects and surfaces in the environment before reaching a listener [7]. Surfaces reflect sound specularly, where the angle of incidence is equal to the angle of reflection (mirror-like reflections when irregularities on the surface are much smaller compared to the wavelength of the sound), and diffusely, where the sound is scattered in several directions (due to irregularities on surfaces in the order of magnitude of the wavelength). They also absorb and transmit some of the incident energy through them. At edges where two surfaces meet or at free-hanging edges, acoustic waves undergo diffraction, where they appear to bend around the edge. Figure 2.1 shows the different kinds of interaction between acoustic waves and surfaces in an environment.

Apart from interacting with surfaces in the room, acoustic waves also interact with the molecules in the atmosphere. Due to shear viscosity, thermal conductivity and molecular relaxation due to vibration, rotational and translational energy of oxygen, nitrogen and water vapour in air, sound undergoes attenuation as it travels in the atmosphere [15]. Air absorption depends on the temperature and relative humidity of air and is frequency

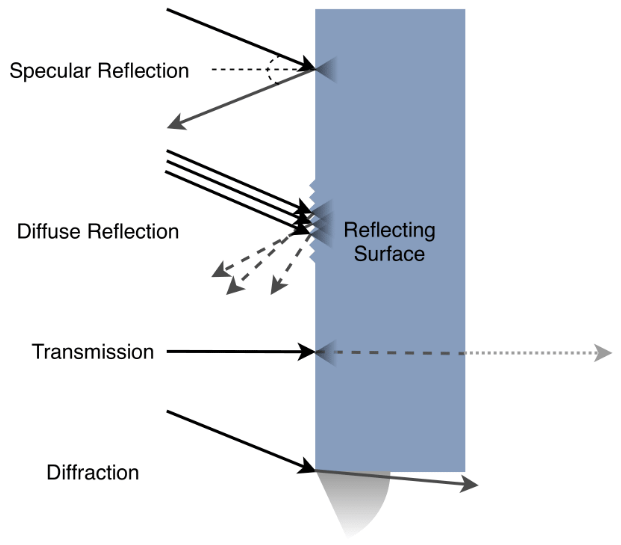


Fig. 2.1: Types of interaction between acoustic waves and surfaces in a room.

dependent. Figure 2.2 shows air absorption coefficients for frequencies in the audible range at 23^oC and a relative humidity of 35%.

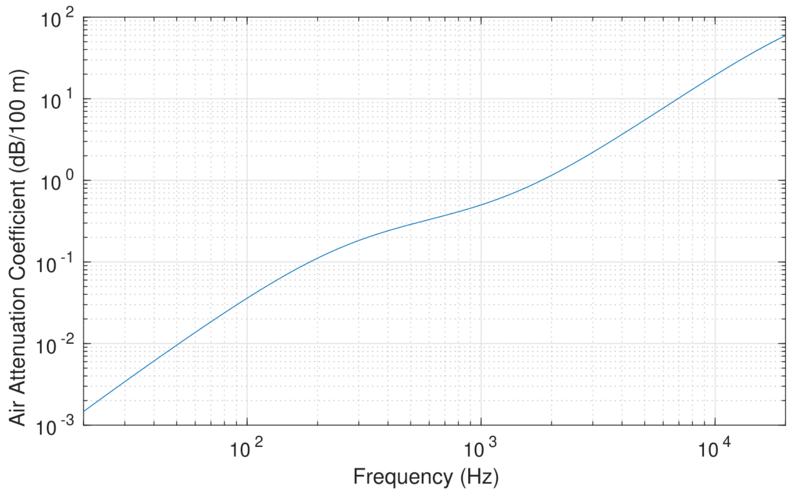


Fig. 2.2: Air absorption coefficients (dB/100m) as a function of frequency of sound (Hz) at 23^oC and 35% relative humidity.

The goal of room acoustic modeling is to provide a realistic representation of the acoustic space by computationally simulating the above mentioned acoustic phenomena. It has found application in architectural design, video games and virtual reality. In architectural design, room acoustic modeling is used to test the acoustic characteristics of the room being designed. In video games and virtual reality applications, auralization of the acoustics in the virtual environment along with spatial rendering of the sound sources [16] enhances experience by helping the user feel completely immersed in the virtual world. Auralization is the process of rendering the acoustic sound field by mathematically or physically modeling the sound field of a source in order to invoke spatial hearing [17] using spatial audio rendering techniques [18].

A room response consists of mainly two parts: the early room response and the late reverberation. The early room response (usually up to 80 ms from the onset of sound) consists of distinct filtered and time delayed copies of the sound from the source that occur due to strong specular reflections and low-order diffraction along with the direct sound, which constitute perceptually significant information [1], helping us in framing an auditory picture of the immediate surroundings. The late reverberation field mostly consists of diffuse reflections and gives us an idea about the size (smaller rooms have shorter reverb tails and larger rooms have longer reverb tails) and sonic colour of the room (bright, dark, muddy, etc.). Since the density of the arrival of reflections increases with time, the late reverberation field usually displays random behaviour due to random phasing between the reflections. Hence, the individual directionalities of these reflections are not perceptually relevant.

Several room acoustic modeling techniques have been proposed in the past to address modeling of one or more of the above mentioned acoustic phenomena. They are broadly classified into two categories: Numerical Acoustics (NA)(wave-based) and Geometrical Acoustics (GA)(ray-based). Numerical acoustic techniques use a wave approach which involves solving the wave equation at sampled locations in the environment with boundary conditions. Finite-Element-Modeling (FEM), Boundary-Element-Modeling (BEM) and Finite-Difference-Time-Domain (FDTD) are some of the numerical techniques used in room acoustic modeling. These techniques model wave-based acoustic phenomena, such as diffraction and interference, to a great degree of accuracy and yield highly reliable results. However, they are computationally very expensive and are usually applied in non-interactive applications where acoustic scenes are pre-rendered, and in architectural acoustics to identify the modes of the room. However, there have been attempts at using such techniques for real-time applications due to the high accuracy of results. Raghuvanshi et al. [8] proposed a technique for real-time simulation of dynamic sound sources in static environments based on a modified-FDTD technique.

BEM and FEM perform computations by frequency, i.e., they compute solutions for each frequency bin for the entire duration of the response, and FDTD techniques perform computations by time steps, i.e., they compute solutions for each time step for the entire frequency range. Both involve computing solutions of huge systems of linear equations [19], which depend on the number of elements being modeled. The sampling density (consequently the number of elements) depends on the wavenumbers of the simulated acoustic waves, i.e., the sampling grid is less dense for lower wavenumbers (lower frequencies) compared to higher wavenumbers (higher frequencies). Figure 2.3 shows a room meshed into triangular elements. Hence, they pose a large memory requirement and their computational load is a function of powers of the number of elements, depending on the technique employed [19]. Moreover, extracting information about the directionality of an arriving wave at the listener is a difficult task. Hence, these techniques are more suited for acoustic simulation at low frequencies, where wave-based phenomena such as diffraction and interference are more prominent and directionality is less significant.

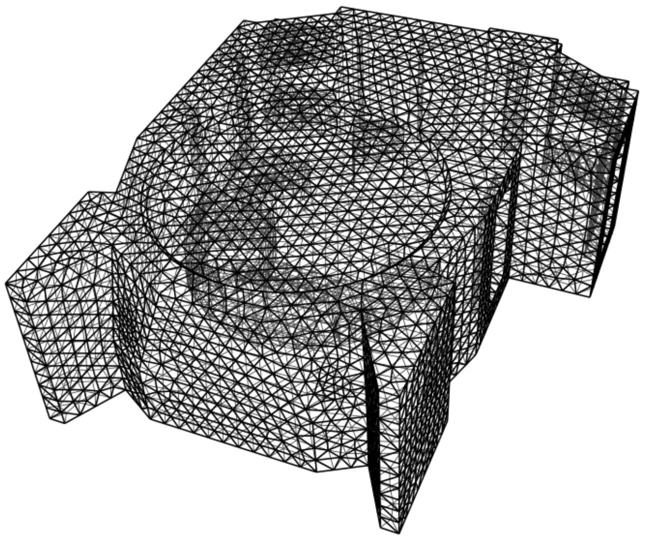


Fig. 2.3: A room meshed into triangular elements for numerical analysis (from [20]).

GA methods, on the other hand, assume rectilinear propagation of sound. This approximation may be inaccurate in the low frequency range but gets asymptotically more accurate in the mid and high frequency ranges. GA techniques are far less computationally expensive compared to NA techniques and can be used in real-time interactive applications, albeit at the cost of accuracy. They share principles with algorithms used in computer graphics, where propagation of light is simulated for realistic rendering of images. Section 2.2 explains some of the popular GA techniques.

2.2 Geometrical Acoustic Techniques

GA techniques are broadly classified into two categories: reflection path-based and surface-based GA techniques.

2.2.1 Reflection Path-based GA Techniques

These techniques use a ray-approach. Some of the common path-based techniques include the image-source method, ray tracing and beam tracing.

Image-Source Method

The image-source method [21,22] is a deterministic method to identify specular reflection paths by reflecting the source against each reflecting surface in the room to find image-sources (points where the reflected rays appear to emerge). These images sources are further reflected against other reflecting surfaces to find image-sources of higher order reflections up to a specified maximum reflection order and are stored in an image tree whose root is the sound source. This method provides a complete list of all possible specular reflection paths up to the specified order. Pulkki et al. [23] have even proposed a method for inclusion of diffraction in the image-source framework. The echogram for the given room configuration is constructed by adding the energies of individual paths with corresponding path delays, along with attenuation and filtering based on the properties of the reflecting surface.

For cuboidal rooms (shoebox-shaped room shown in Fig. 2.4) with no obstructing surfaces, the image-source method gives the exact solution of all specular reflections. For more complex room geometries, many image-sources may be invalid due to obstruction or physical incoherence. Figure 2.5 shows the image-sources for a source in a concert hall along with the image tree. We can see that only a very small number of these image-sources are actually valid. In such complex geometries, additional checks need to be performed in order to avoid computing invalid image-sources [22]. In spite of this pruning, a major drawback of the image-source method is that the image tree grows exponentially with increasing reflection order, thus posing a large memory requirement for higher reflection orders. Hence, the image-source method is suited for early specular reflections only.

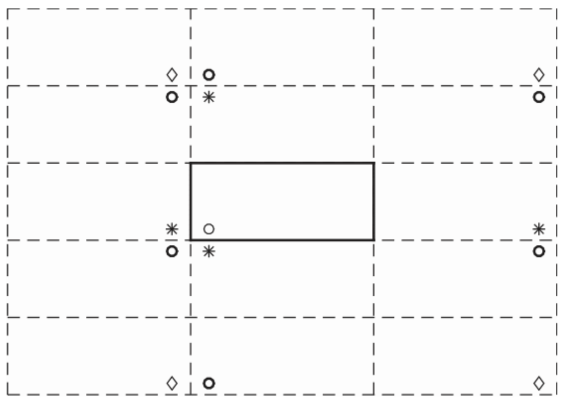


Fig. 2.4: Example of image-source method in a shoebox-shaped room in 2-D. ◦ is the source, * are the first-order image-sources, **o** are the second-order image-sources and ⋄ are the third-order image sources (from [7]).

Ray Tracing

Ray tracing [24] is another common technique, which involves projecting a large number of rays from the source, tracking them through reflections and registering rays that strike the listener. Ray tracing is a stochastic method that converges towards the exact solution provided by the image-source method for specular reflections with increasing density of rays being projected. The directions of rays from the source can either be predetermined by a known directivity-based distribution or can be chosen through Monte Carlo sampling [7]. What makes this technique attractive is the ease with which diffuse reflections [25,26,27] and diffraction [28,29] can be incorporated in the model to improve accuracy of reverberation time. Although this technique has proven to yield satisfactory results and is probably the most preferred technique, a major drawback is the systematic errors that it produces due to finite number of rays being traced [30,31].

Size of the listener is also a significant cause for unreliable results [7]. When the listener is point-sized, the probability of a ray striking the listener is very low. In order to overcome this issue, a listener with finite volume may be used. However, if the size of the listener is too large, invalid paths may be registered, causing the result to be unreli-

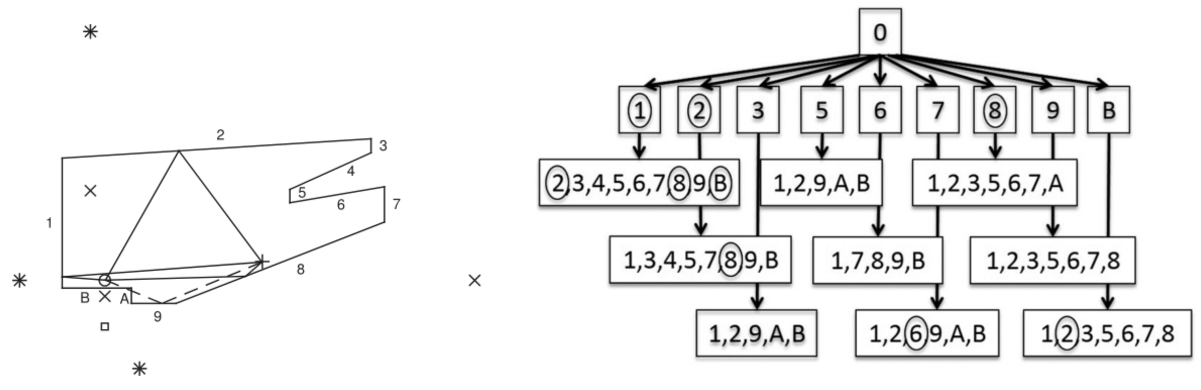


Fig. 2.5: On the left, the figure shows the image-sources constructed for the source (∘) and reflection paths to the listener (+) in a concert hall (reflecting surfaces marked 1-9 and A-B). The valid image-sources are marked by $*$ and the image-sources that are not visible to the listener are marked by $\times$. The image-source marked by a □ is occluded (occluded path shown by a dotted line) since surfaces A and B are in the way. On the right, we see the image-tree constructed for the source location and the room geometry, with only the valid image-sources circled (from [7]).

able again. Vorländer [32] describes a solution to this problem where the volume of the listener increases as a function of distance travelled by the ray.

Beam Tracing

Beam tracing can be viewed as a solution to the above problem, where instead of tracing individual rays, bundles of rays defined by bounding frusta are traced. This is also known as frustum tracing in which the spherical surface of the source is tessellated into flat polygonal sections, which form the cross-sections of the beams, such that the beams cover roughly equal solid-angles [7] (Fig. 2.6). When these beams are incident on reflecting surfaces, they may split depending on whether the beam is partially or fully incident on the surface, and then traced further. This approach is very similar to ray tracing with rays being replaced by beams, thereby reducing the number of entities to be traced, but can have an increasing computational cost due to beam splitting. Beam tracing with conical beams (circular cross-section) have also been proposed [32,33] but face problems of either completely missing paths or registering paths more than once due to imperfect tessellation of the surface of the source into circular regions.

Another version of beam tracing is an optimization of the image-source method [1,9, 34]. In this method, initially, the first order image-sources are computed for all visible

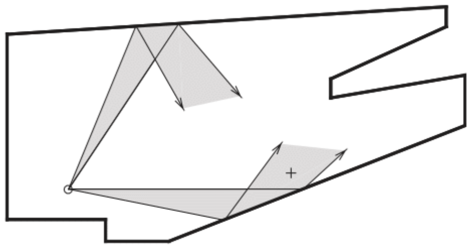


Fig. 2.6: Frustum tracing for a source in a concert hall (from [7]).

reflecting surfaces and beams are constructed using these image-sources as apexes and the reflecting surfaces forming cross-sections. These beams are used to narrow down the search space for higher order image-sources to only those surfaces that intersect the beams. This prevents the number of image-sources from growing exponentially, hence resulting in a narrower image tree (Fig. 2.7). The beam tracer discussed in this thesis belongs to this category of beam tracers. Section 2.3 explains this method in detail. Apart from modeling specular reflections, Funkhouser et al. [9] and Tsingos et al. [35] have also presented methods in which diffraction can be modeled within the beam tracing framework.

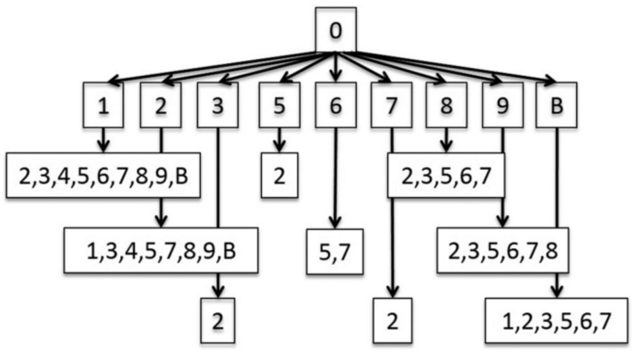


Fig. 2.7: Beam tree constructed for the concert hall scenario in Fig. 2.5. The tree obtained from the beam tracing algorithm is narrower compared to the tree obtained from the conventional image-source method (from [7]).

2.2.2 Surface-based GA Techniques

Radiosity and Acoustic Radiance Transfer (ART) are GA methods that are surface-based techniques for room acoustic modeling. They are multi-pass techniques [7] where energy propagation is from the source to the reflecting surfaces in the first pass and energy propagation is between the reflecting surfaces in all other passes, except the last pass. Only in the very last pass, the listener comes into the picture, hence making most of the computation independent of the listener which can be computed ahead of time. Both radiosity and ART assume ideally diffuse reflections and are used to estimate reverb time and energy decay in rooms.

Radiosity has been extensively used in computer graphics for global illumination in diffusely reflecting rooms. This method makes use of angle independent bidirectional reflectance distribution functions (BRDF) as opposed to more complex angle dependent ones that are used in ART, hence is computationally cheaper but gives only a crude approximation. BRDFs are functions that give the ratio of incoming and outgoing energy as a function of incoming and outgoing angles [7]. Moreover, in ART, the reflecting surfaces are split into smaller patches in order to model reflections with greater spatial resolution. Hence, ART yields results with greater directive information, but with increased computation and memory requirements.

Surface-based techniques are similar to numerical methods such as BEM but model acoustic energy rather than acoustic pressure. Due to the lack of phase information assuming wave-based effects are negligible, these techniques are suited for simulation at higher frequencies. Furthermore, since energy accumulation at reflecting surfaces yields an approximation of the energy in space and time, these techniques are more suited for computation of late reverberation field.

Since each of the above mentioned techniques have their own merits and perform better in modeling some acoustic phenomena than others, hybrid methods have been proposed that make use of more than one technique to address different parts of the room response, achieving a result that is coherent overall (some hybrid techniques are presented in [7]). Siltanen et al. [19] present a review of acoustic modeling techniques and proposes hybrid solutions where numerical methods are used for low frequencies, reflection path-based methods are used for early room response in mid- and high-frequency ranges and surface-based methods are used for late reverberation in high frequencies. However, the paper does not comment on the cross-over between these techniques and how their results can be stitched together seamlessly. Figure 2.8 shows the different methods and their suitability for various time and frequency ranges.

Savioja et al. [7] also present a comparison of the different GA techniques on the basis of memory requirement, computation time and their suitability for modeling different

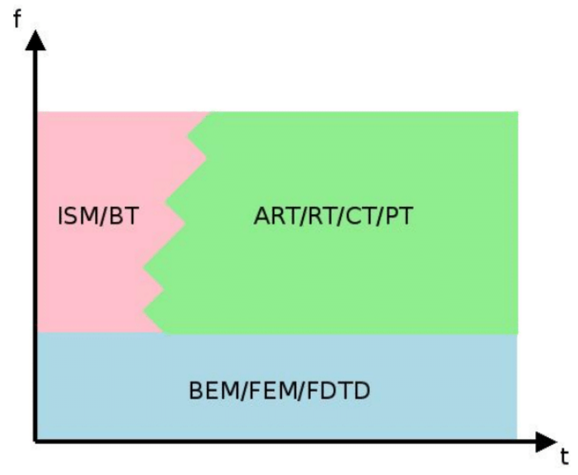


Fig. 2.8: Suitability of modeling techniques for different regions in the time-frequency plane. ISM=Image-source method, BT=Beam tracing, ART=Acoustic radiance transfer, RT=Radiosity, CT=Cone tracing, PT=Particle tracing(Ray tracing), BEM=Boundary element modeling, FEM=Finite element modeling, FDTD=Finite difference time domain (from [19]).

parts of the room response. Among the path-based techniques that model the early room response containing the most perceptually relevant reflections, ray tracing and beam tracing seem to be the most attractive options. Although ray tracing has a lower memory requirement, the computation time is slightly higher, making beam tracing more favourable for modeling early reflections in real-time applications. However, state-of-the-art beam tracing algorithms [1, 12] are either only listener-dynamic, i.e., change in the listener position does not demand complex computations for regenerating valid paths, or suitable only for static geometries unlike the ray tracing algorithm. The aim of this thesis is to reduce the computation time of the beam tracing algorithm, hence making it more source-dynamic. Section 2.3 explains the beam tracing algorithm in detail.

2.3 Beam Tracing

In the context of beam tracing as an optimization of the image-source method, beams contain information about regions of the virtual environment that are visible to the source through reflections. Starting from the source, these beams are traced through several orders of reflection by successively reflecting them by the intersecting surfaces/polygons to build a complete map of visibility. This visibility map is known as the beam tree. Beam trees are similar to the image-tree mentioned in Sec. 2.2.1. In this section, we will mainly focus on the implementation of the beam tracing algorithm by Laine et al. [1][1].

A beam is geometrically defined by the source/image-source at the apex and the surface of reflection forming a cross-section. For example, in Fig. 2.9, the reflected beam B_r is defined by the image-source S' and the four bounding surfaces which contain the edges of the reflecting surface P_r. Tracing these beams through reflections in the room involves identifying surfaces in the room geometry that intersect them, which is a computationally expensive process. The geometrical operations themselves become simpler when the surfaces are polygonal, planar and convex, i.e., all internal angles of the polygon are less than 180^o. Hence, in Laine's implementation[1], all planar surfaces in the room are broken down into convex polygonal planes and all curved surfaces are tessellated piece-wise into flat convex polygons and added to the definition of the room geometry. This ensures that the regions of beam-polygon intersections are also polygonal, resulting in convex reflecting beams for all reflections and avoiding complex geometrical operations altogether.

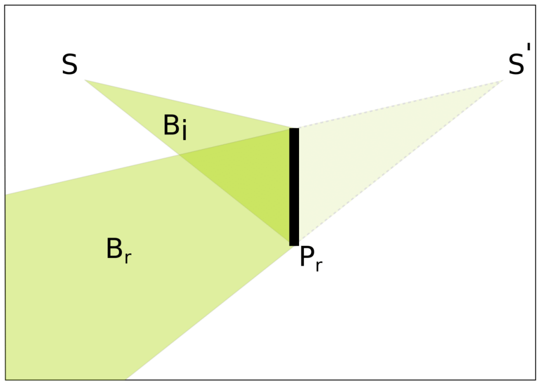


Fig. 2.9: Here, the beam B_i is reflected by the polygon P_r. The reflected beam B_r is defined by the image-source S' of the source S with respect to P_r and the edges of P_r.

The complexity of searching for beam-polygon intersections grows linearly with increasing number of polygons in the room geometry. Hence, for room geometries of increasing complexity, this task becomes more and more impractical. In order to accelerate this process, the room geometry is spatially subdivided and the polygons that make up the room geometry are stored in a Binary Space Partitioning (BSP), facilitating efficient searches.

[1]The implementation can be found in https://users.aalto.fi/ laines9/publications/laine2009aa_code.zip.

2.3.1 Binary Space Partitioning (BSP) Tree for Efficient Beam Tracing

The BSP tree data structure used in Laine's implementation is a binary tree whose nodes contain split planes that are axially aligned (the split planes are parallel to axial planes). Effectively, these split planes demarcate regions of the virtual environment such that the total areas of the polygons on either side of the split plane are similar. The split planes, together, define subdivisions of the room geometry as Axis Aligned Bounding Boxes (AABB), which are cuboidal regions enclosing polygons. Figure 2.10 shows a simple box split into smaller AABBs by split planes and the corresponding BSP tree.

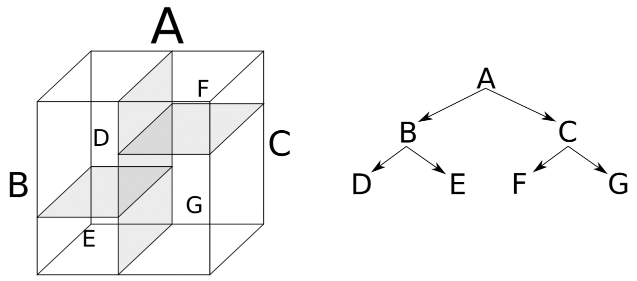


Fig. 2.10: On the left, the whole cube A is split into two by the vertical split plane forming two child AABBs, B and C. The children are further split by horizontal split planes into D, E, F and G. The split planes are shown as shaded flat quadrilaterals. The corresponding BSP tree is given on the right.

Construction of the BSP tree is a pre-processing step which is done once for the given room geometry. Each time the room geometry changes, a new BSP tree is built. Algorithm 1 explains the process of construction of the BSP tree.

2.3.2 Beam Tree Construction

Once the BSP tree is built, a beam tree is constructed for the given source location, which holds the information about regions of visibility from the source. Beam tree construction is also considered a pre-computation step that is processor intensive and is done once for a given source position and room geometry. Every time either of them changes, a new beam tree is computed. Hence, the computational cost of the beam tree construction process restricts how quickly the source can move, i.e., the algorithm is not source-dynamic. The stopping condition is usually a maximum reflection order or a maximum distance of beam traversal. In Laine's implementation, the beam tree is constructed in a depth-first manner (starting from the root and tracing all the beams along one branch completely

Algorithm 1 BSP tree construction

```
Initial: $poly$ := polygons in the scene
procedure CONSTRUCT_BSPTREE($poly$)
    $node$ := new BSP tree entry
    [$node.split\_position$, $node.split\_axis$] = OPTIMAL_SPLIT($poly$)
    if no optimal split then
        $node.polygons$ := $poly$
        return $node$
    end if
    $left\_poly$ := polygons to the left of split plane (complete or partial)
    $right\_poly$ := polygons to the right of split plane (complete or partial)
    $node.left\_child$ := CONSTRUCT_BSPTREE($left\_poly$)
    $node.right\_child$ := CONSTRUCT_BSPTREE($right\_poly$)
    return $node$
end procedure
```

before backtracking) and beam tracing is terminated on reaching a maximum order of reflection. The basic beam tracing algorithm as implemented by Laine is given in Algorithm 2.

Each beam tree node contains the reflecting polygon and the image-source position for that reflection. The beam tracing algorithm given in Algorithm 2, in essence, starts with the first order image-sources corresponding to all surfaces in the room, regardless of whether they are actually visible to the source or not (in Algorithm 2, starting

Algorithm 2 Beam tracing

```
Initial: $source$, $beam = NULL$, $order = 0$ and $parent = NULL$.
procedure BEAMTRACE($source$, $beam$, $order$, $parent$)
    if $order \geq max\_order$ then                         ▷ Check stopping condition
        return
    end if
    $intersecting\_polygons$ = BEAM_CAST($beam$, $BSP\_Tree$)
    for $int\_poly \in intersecting\_polygons$ do
        $image\_source$ = MIRROR($source$, $int\_poly$)          ▷ Find image-source
        $new\_beam$ = BEAM($image\_source$, $int\_poly$)         ▷ Construct new beam
        $parent.new\_child \leftarrow$ NODE($image\_source$, $poly$)
        BEAMTRACE($image\_source$, $new\_beam$, $order$+1, $parent.new\_child$)
    end for
end procedure
```

with a $NULL$ beam gives all first-order image-sources). Beams, which are defined by the image-sources and their respective reflecting polygons, are reflected by the polygons that intersect with them and are traced further. The beam tracing algorithm proposed by Funkhouser et al. [34] uses a cell adjacency graph that helps in projecting beams from one region towards adjacent regions only, thereby preventing unnecessary computation of beams to regions that are occluded. The cell adjacency graph contains nodes which represent the different regions of the geometry stored in the BSP tree, with links between adjacent regions for every surface that separates them. Figure 2.11 shows the cell adjacency graph for a simple room model. Laine's implementation of the beam tracing algorithm skips occlusion tests during the pre-computation stage, hence does not require an adjacency graph.

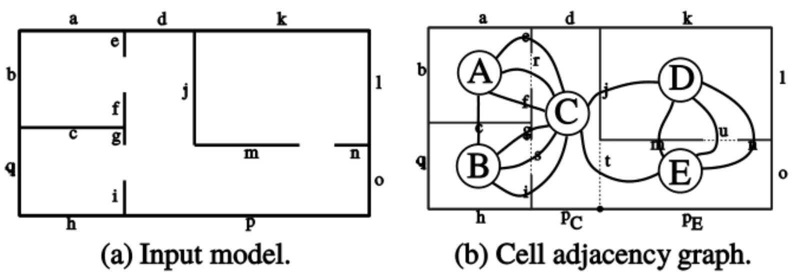


Fig. 2.11: a) Shows the input model and b) shows the corresponding cell adjacency graph constructed for the given input room model. The nodes A-E represent regions in the BSP tree and the links between them represent the adjacency of these regions through separating surfaces. This graph is used for avoiding beam computations for occluded regions (from [34]).

Polygons that intersect the beam under consideration are identified by performing frustum culling (a common technique used in computer graphics to identify objects in the viewing frustum) with the AABBs, given by the split planes stored in the BSP tree. The BSP tree greatly helps in accelerating the process of successive frustum culling to identify intersecting polygons. Image-sources are computed only for these intersecting polygons, hence preventing exponential growth of the beam tree.

Despite the acceleration achieved by the use of BSP trees, the process of beam tree construction is still computationally expensive and can be used only for simple room geometries for low-orders of reflection at interactive rates of moving sources [1]. Antonacci et al. [10,11] (in context of 2-dimensional beam tracing) and Marković et al. [12] (in context of 3-dimensional beam tracing) proposed techniques based on precomputed look-up tables that contain mutual visibility of surfaces in the room. In [12], mutual visibilities are computed by transforming rays, beams and surfaces to 5-dimensional Plücker space where complex computations become simple *"iterative intersection of linear subspaces"*. This has proven to speed-up the beam tree computation process, facilitating tracing of a large number of beams at interactive rates with a highly dynamic source. However, building the look-up table is a computationally very expensive process (could take a few minutes to hours [12]) which is done as a pre-processing step, hence is not suitable for environments with dynamically changing room geometries (ex. a virtual environment where doors/windows can be opened/closed and objects in the room can be moved around).

Funkhouser et al. [9] proposed three techniques to accelerate the beam tracing process: 1) Prioritized beam tracing, 2) Bi-directional beam tracing, and 3) Amortized beam tracing. The priority of a beam in the prioritized beam tracing is given by the sum of the distance travelled by the beam up to the reflecting surface and the distance of the listener from the surface. Since the shortest distance between the listener and the surface is considered, it does not guarantee that the reflected beam contains a valid path from the source to the listener due to possible occlusions by other surfaces. Moreover, since the beam tracing process depends on the listener location, the algorithm is less listener-dynamic and is more expensive when there is more than one listener since a separate beam tree must be computed for every source-listener pair.

The bi-directional beam tracing algorithm constructs two beam trees, one starting from the listener and the other starting from the source, up to a low reflection order (which is faster than computing one beam tree up to high-orders of reflection) and combines them by identifying overlapping beams. Although bi-directional beam tracing accommodates inclusion of diffraction, it fails for the same reasons as the prioritized beam tracing (less listener-dynamic and requires separate beam trees for each source-listener pair) and also due to additional computations for detection of overlapping beams.

In the amortized beam tracing, a conservative beam tree is constructed by tracing polyhedral beams from a region around the source that results in an over-estimated tree containing possible reflection paths for future source positions. The concept of amortized beams, given Laine's optimizations to the path validation process (explained in Sec. 2.3.3), could greatly improve the overall computation speed of the beam tracing algorithm. However, in this thesis, we only present optimizations for beam tracing from the exact source location, although amortization can be included with minimal changes.

2.3.3 Path Validation

The beam tree obtained gives the possible sequences of reflections from the source and does not provide exact information about valid paths from the source to the listener. The validation of paths (identifying unobstructed reflection paths between the source and the listener) is performed by a ray tracer that traces rays from the listener, starting at each beam tree node, traversing all the way to the root of the beam tree, which is the source. The path validation algorithm is given in Algorithm 3.

Algorithm 3 Path Validation

```
Input: Beam tree, source and listener positions
Output: $valid\_paths$
for $node \in beam\_tree$ do
    $src \leftarrow node.image\_source$
    $target \leftarrow listener\_position$
    $valid \leftarrow$ True
    $cur\_path$ := empty list                                ▷ List of points of the path
    while $node \neq root$ do
        $cur\_path$.APPEND($target$)
        $ray\_1$ := RAY($target$, $node.image\_source$)
        if $ray\_1$ does not intersect $node.polygon$ then
            $valid \leftarrow$ False
            break
        end if
        $isect$ := intersection of $ray$ and $node.polygon$
        $ray\_2$ := RAY($target$, $isect$)
        if $ray\_2$ intersects other polygon then            ▷ Check for obstruction
            $valid \leftarrow$ False
            break
        else
            $target \leftarrow isect$
            $node \leftarrow node.parent$
        end if
    end while
    if $valid$ = True then
        $cur\_path$.APPEND($source\_position$)
        $valid\_paths$.ADD($cur\_path$)
    end if
end for
```

Path validation is performed each time there is a change in the listener position. Even though the process of going through the entire tree node-by-node is not as computationally expensive as the beam tree construction itself, it can be further optimized since most of the nodes in the tree result in invalid paths due to obstruction or rays not intersecting the reflecting surfaces. This would improve the rate at which the listener can move, making the algorithm highly listener-dynamic. Laine et al. [1] proposed two techniques to accelerate the path validation step, where redundant computations that give negative results are avoided, thereby achieving a speed-up by several times.

A path could be invalid in the following conditions:

a) The image-source and the target are on either side of the reflecting polygon but the ray between them does not pass through the polygon.

b) The image-source and the target are on the same side of the reflecting polygon.

If the path validation fails at any given node in the beam tree, paths from nodes lower in the same branch are also likely to fail. Two techniques were proposed to help identify nodes for which path validation can be avoided: 1) fail-plane optimization, 2) skip-sphere optimization.

Fail-plane Optimization In this optimization technique, a plane is constructed for every node in the beam tree to help determine whether or not to continue path validation through that node. This plane is known as the fail-plane. A path through a node is valid if the target is behind the fail plane, and invalid if the target is in front of the fail plane. This can be determined by computing the dot product of the target position and the normal to the fail plane. If the dot product has a positive sign, the target is in front of the fail plane and the path is invalid. If the path is invalid due to condition a) given above, the fail plane is that plane of the beam originating from the image-source, whose perpendicular distance to the target is the smallest (see Fig. 2.12a). For condition b), the reflecting surface with its normal pointing towards the image-source of the node is the fail plane (see Fig. 2.12b).

This fail plane is propagated down the branch to all the nodes by mirroring the plane at each node by the reflecting surface. Thus, before starting the path validation process from a node, it would be sufficient to check if the listener is in front of the fail plane to reject the path. Further explanation of this optimization is given in [1].

Skip-Sphere Optimization In this optimization technique, nodes of the beam tree are grouped into buckets of a specified size (Laine's implementation uses a bucket size of 16). For each bucket, a sphere centred at the listener location with radius equal to its distance from the nearest fail plane is defined. As long as the listener is within this sphere, paths from all the nodes in the bucket would be invalid and path validation for the entire bucket can be skipped. If the listener moves out of the skip sphere, it can be guaranteed that at least one path in the bucket has changed, hence all the nodes are validated. Figure 2.12c illustrates this with an example. Further explanation and implications are given in [1].

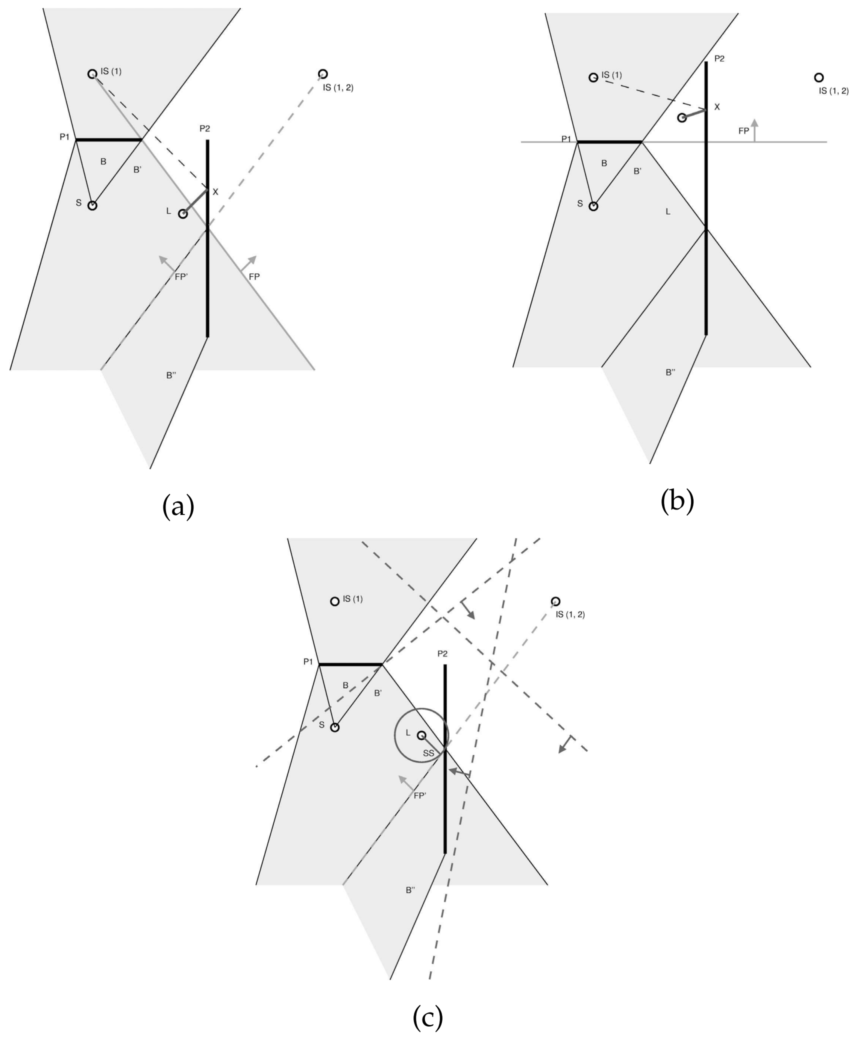


Fig. 2.12: Optimizations to the path validation step proposed by Laine et al. [1].

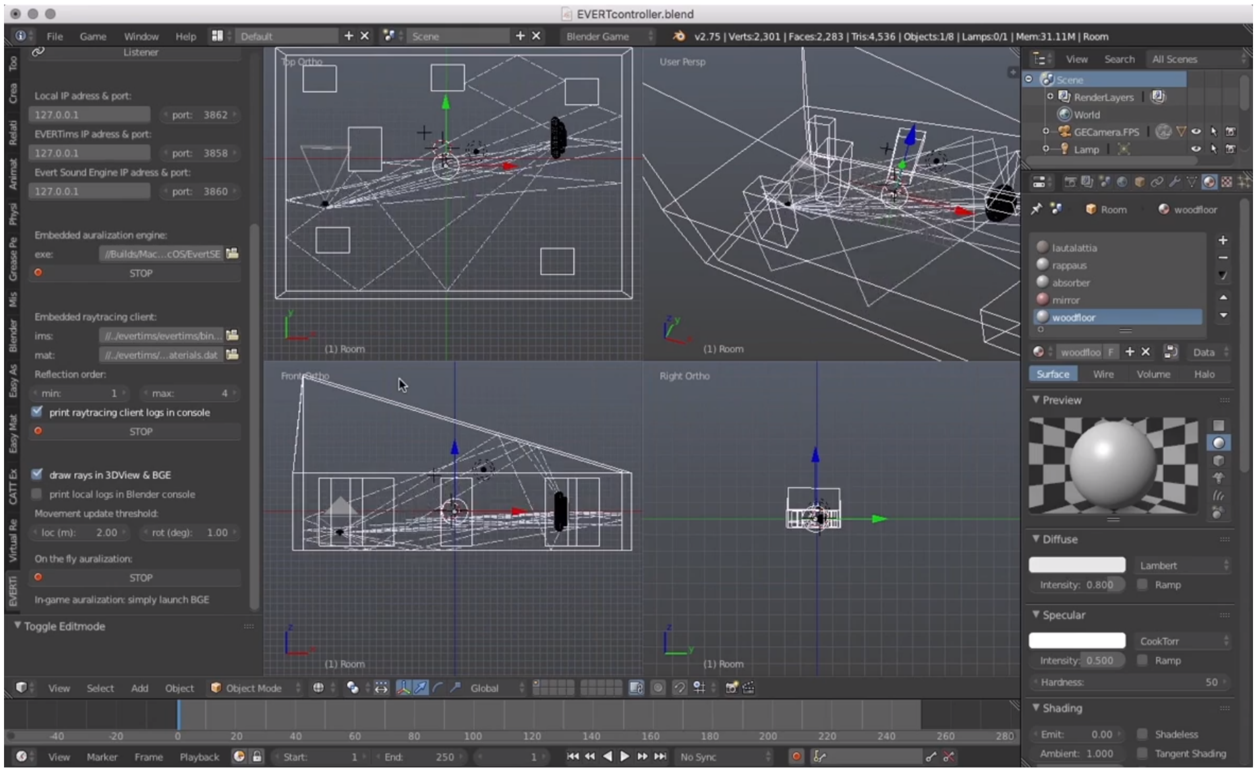
The two optimization techniques drastically improve the speed of the path validation step by reducing the number of beam tree nodes to validate, thereby making it possible to quickly compute reflection paths whenever a listener moves, without performing heavy computations. The ability to compute reflection paths for a moving listener at interactive rates makes the beam tracing algorithm listener-dynamic. Laine et al. [1] provide results of comprehensive timing tests conducted for rooms with various number of surfaces and various levels of complexity. Overall, the two techniques together provide up to 90x computational gain.

2.4 Beam Tracing in a Real-time Application: EVERTims

EVERTims by Noisternig et al. [13] is an open-source framework for real-time room acoustic simulation and auralization that makes use of the beam tracing algorithm for early specular reflections. The package contains three different sub-systems: 1) VirChor (Virtual Choreographer) [36], a real-time 3D graphics rendering engine for control of the geometric room model, the source and the listener, 2) EVERT room acoustic modeler, a beam tracer for finding early specular reflection paths between the source and the listener in the room, 3) Spatial rendering and auralization engine for audio rendering in ambisonics [37] over loudspeakers and binaural audio [38] over headphones. The three subsystems communicate with each other using the Open Sound Control (OSC) protocol to transfer information about source-listener positions and reflection paths. In the latest version of EVERTims [14] VirChor is replaced with Blender [39], an open-source 3D modeling software. Figure 2.13 shows a room with a source and a listener and the valid paths between them.

The beam tracer in this software package uses the algorithm presented by Laine et al. [1] (explained in detail in Sec. 2.3), which builds a conservative beam tree and performs path validation to identify valid specular reflection paths. For auralization in real-time, specular reflection paths are identified and sent to the auralization engine in bursts, grouped by reflection order. The EVERTims application provides a parameter to specify the minimum required order of reflections, up to which the beam tracer first constructs the beam tree and sends the valid paths. Once this minimum order is reached, the beam tracer sequentially performs beam tracing for every higher reflection order up to a maximum specified order of reflections in the background. On completion of beam tracing up to each intermediate order, only the new paths are sent to the auralization engine. This technique is known as *iterative order refinement*. It is most common to use a minimum reflection order of 1 and a maximum order of reflection as required by the user for quick rendering of at least the bare-minimum reflection paths as the source moves in real-time. Since EVERTims uses a depth-first beam tracer, a new beam tree is built from scratch for each order of reflection to achieve iterative refinement of the obtained reflection paths. Hence, this adaptation of the depth-first beam tracer may be referred to as a pseudo-breadth-first beam tracer, whose overall performance resembles that of a breadth-first beam tracer, providing valid reflection paths by reflection order, with the beam tracing within each order being performed in a depth-first manner.

The auralization engine uses delay lines for the early reflection paths along with octave-band filters for shaping their frequency responses due to material absorption. The material absorption coefficients are retrieved from a database which contains coefficients of various common materials such as brick, wood, glass, concrete, plaster, etc. for 10 octave bands in the audible frequency range. The octave band filter is implemented by cascading first-order butterworth low-pass filters in order to achieve a constant passband response.


EVERTcontroller.blend

Local IP adress & port: 127.0.0.1 port: 3862
EVERTims IP adress & port: 127.0.0.1 port: 3858
Evert Sound Engine IP adress & port: 127.0.0.1 port: 3860

Embedded auralization engine:
exe: //Builds/Mac...cOS/EvertSE
STOP

Embedded raytracing client:
ims: //../evertims/evertims/bin...
mat: //../evertims/...aterials.dat
Reflection order: min: 1 max: 4
[✓] print raytracing client logs in console
STOP

[✓] draw rays in 3DView & BGE
[ ] print local logs in Blender console
Movement update threshold: loc (m): 2.00 rot (deg): 1.00
On the fly auralization:
STOP
In-game auralization: simply launch BGE

Fig. 2.13: Screenshot from EVERTims on Blender

Although such a filter suffers from variable group delay for each octave band, the use of low-order filters makes the group delay less perceptible, but giving an overly smooth approximation of the frequency response of materials.

Apart from handling the early reflection paths, the auralization engine also accounts for the late reverberation by modeling the diffuse reflections stochastically with frequency dependent reverberation times estimated from the volume of the room and the absorptive area of surfaces in the room provided by the room modeler, using the Sabine formula [40]. The late reverberation field is auralized using a Feedback Delay Network (FDN) [4] with 16 feedback channels, using the early reflection paths responses as the input. The overall room acoustic modeling technique used in this package can be considered a hybrid method since the early room response is modeled deterministically (beam tracing) and the late reverberation is modeled stochastically (FDN).

In the latest version of EVERTims [14], the early reflection paths are encoded into 3rd order ambisonic channels to achieve high spatial resolution and the diffuse late reverberation is encoded into 1st order ambisonic channels. For auralization through headphones, these ambisonic channels are transformed into two-channel binaural audio using a virtual speaker technique [41]. In order to enhance the directionality of the direct path, the direct path bypasses the ambisonic encoding step and is directly encoded in binaural.

2.5 Challenges with Beam Tracing

The most expensive process in the beam tracing algorithm is the beam tree construction. In an interactive application, each time a source moves, a new beam tree must be constructed since the visibility map of the room changes for each location of the source. As previously explained, it involves beam casting to identify intersecting polygons, reflecting image-sources to find new image-sources, constructing new beams, etc. several thousand times depending on the complexity of the environment. The number of polygons in the room geometry dictates the size of the beam tree, which in turn affects the computation time. In spite of the beam tracing algorithm yielding narrower trees compared to the image-source method, beam trees still grow exponentially with increasing reflection orders. Performing occlusion tests during beam tracing helps in further reducing the size of the beam tree but the advantage of building a conservative tree with faster validation techniques outweighs the gains of occlusion testing due to the high added cost and additional data structures for occlusion tests [1]. Hence, beam tracing has proven to be useful only for simple to moderately complex room geometries up to a low-order of reflections for slow moving sources.

Although several works in the past [9,10,11,12,34] have attempted to address the issue of high computational cost of the beam tree construction phase, the solutions proposed either lead to loss of tolerance to listener movement, add further costs in multi-source multi-listener systems or work only in static geometries (see Sec. 2.3.2). The "iterative order refinement" solution proposed in [13,14] for the use of beam tracing in real-time is an excellent workaround for the problem. It involves performing beam tracing in steps to have the most perceptually relevant reflection paths (lower order reflections) auralized quickly in response to change in source position and then gradually refine the solution to improve the accuracy.

However, the use of a depth-first beam tracer in an iterative refinement framework leads to a major drawback of increased overall cost while tracing up to high reflection orders due to repeated computation of parts of the beam tree. This is due to the fact that with a depth-first beam tracer, in order to incrementally compute the beam tree, there is no easy way to find the beams in the leaf nodes of the tree at a given order without retracing beams from the start. This redundancy is purely a consequence of the design choice of using a depth-first beam tracer. For such a framework that uses an iterative refinement approach, a breadth-first beam tracer that traces beams order-by-order without having to perform redundant computations (traces all beams of the current reflection order before moving on to the next order of reflection) or a best-first beam tracer that traces beams based on some priority score (as described in [9]) would be more suitable since both techniques have mechanisms to store beams until they are traced. In spite of having higher computational costs and greater memory requirements to save beams, both the breadth-first and the prioritized beam tracer would be more efficient due to no redundant computations.

For example, in a cube shaped room with the source at the center, the first level of the beam tree contains 6 nodes, each corresponding to a surface and all nodes, except the root, have 5 child nodes (corresponding to all polygons except the one that reflects the beam). Hence, the second level of the beam tree contains 30 nodes, the third level contains 150 nodes, fourth level contains 750 nodes, and so on. Since the EVERTims software package uses a depth-first beam tracer that uses recursive functions, hence losing track of its leaf nodes, for every order of reflection between 1 and 5, a new beam tree must be constructed. Hence, a total of 6,851 nodes are computed when the final beam tree of the 5^{th} order consists of only 5,686 nodes. This means that close to 20% of the beams are retraced and this redundancy goes up to 25% for higher orders of reflection. In more complex geometries, the redundancy will only be higher.

Looking closely at the process of beam tree construction, we can see that the order in which beams are traced is rather arbitrary and depends on the order in which the reflecting polygons are stored in memory. Many beams have a very small chance of yielding significant higher order reflection beams and paths of higher energies and only some beams may be worth tracing further. Tracing these beams may improve the chances of finding valid reflection paths earlier in the beam tracing process, therefore, having to build smaller trees. Funkhouser et al. [9] recognize this and propose a prioritized beam tracer that traces beams that are close to a listener in the hope that valid paths to the listener would be recognized earlier. However, as explained in Sec. 2.3.2, since the priority depends on the shortest distance from the point of reflection of the beam and the listener without considering occlusions, existence of valid paths to the listener cannot be guaranteed. Also, the solution being heavily dependent on the listener position, it is not listener-dynamic. Besides having to compute a new beam tree when a source moves, we would also have to compute a new beam tree when the listener moves.

In this thesis, we propose a method to reduce the size of the beam tree constructed by prioritizing beams based on the energy that they carry. The goal is to build smaller beam trees without depending on the listener position in order to retain the tolerance for listener movement and also have an adaptive stopping condition which can be made use of in an iterative refinement framework for real-time path finding. We present a comparison of the proposed prioritized beam tracer with the conventional depth-first and breadth-first approaches, and also with the mixed approach (pseudo-breadth-first) used in the EVERTims software package in Chapter 4.

Chapter 3

Prioritization of Beams

In this chapter, we present the various factors that affect the energy carried by a beam, such as width of the beam, distance travelled by the beam and absorption by the reflecting surfaces, and discuss how these factors are computed to assign priorities to beams. We will not be covering directional radiation of the source as a factor for prioritization as it is out of the scope of this thesis, although it is presented as a possible factor and will be dealt with in a future project. We also explore different termination conditions for prioritized beam tracing and propose pausing conditions using adaptive mechanisms.

3.1 Factors That Affect The Energy of a Beam

In the context of beam tracing, beams essentially carry fractions of the total energy radiated by the source. The amount of energy carried by a beam depends on the following factors:

1. **Width of the beam:** Assuming the source radiates equal energy in all directions, i.e., the source is omnidirectional, the amount of energy radiated within a beam can be determined by the width of the beam or the solid-angle it subtends at the source. Wider beams carry more of the energy emitted by the source than narrower beams.

2. **Air-absorption:** As beams travel in air, they undergo attenuation due to absorption by molecules in the atmosphere. Longer beams lose more energy due to air-absorption, hence carry less energy compared to shorter beams.

3. **Material-absorption:** When beams are incident on reflecting surfaces, some of the energy is absorbed and the rest is reflected (assuming no transmission takes place through the surface). Beams reflected by surfaces made of more absorptive materials carry less energy.

4. **Directivity of the source:** Sound sources in the real world radiate different amounts of energy in different directions. Depending on the direction in which a beam is emitted, it could carry more or less energy. This factor, however, is not covered as the computations involve complex 3-D mathematical functions and is out of scope of this thesis.

In the beam tracing algorithm, on completion of one beam tracing step (tracing one beam, identifying the reflecting polygons and adding them to the beam tree), the algorithm can choose a leaf node from the beam tree constructed so far, through which the beam tracing process is continued, growing the beam tree further. In a *depth-first* beam tracer, the algorithm chooses the reflected beam from the first reflecting surface that is identified to intersect the previous beam. In a *breadth-first* beam tracer, the algorithm chooses the first beam from the list of untraced beams of the same reflection order as the previously traced beam. In both cases, the choice of the next beam to trace is rather oblivious to the physical configuration of the environment and is more dependent on the order in which the polygons are stored in the BSP tree. In this thesis, we try to influence the choice of the next beam to trace using the above mentioned factors that affect the energy carried by a beam. This is done in order to guide the growth of the beam tree in a way that makes more physical sense, maximizing the likelihood of finding valid reflection paths earlier, given no knowledge about the listener position (in order to maintain dynamic movability of listeners).

Section 3.2 explains in detail how priority factors are computed based on the above mentioned factors.

3.2 Computation of the Priority Factors

3.2.1 Beam-Width Factor

The width of the beams help in prioritizing beams that are wider and carry more of the emitted energy. The beam-width or the solid-angle it subtends at the source/image-source (will be referred to simply as image-source henceforth since the actual sound source can be considered as the zeroth-order image-source) depends on the surface area of the reflecting polygon, the distance of the reflecting polygon from the image-source and the orientation of the reflecting polygon with respect to the image-source. Figure 3.1 shows an example comparing the beam widths of reflecting surfaces of different surface areas, distances from the image-source and orientations. In the figure, although polygons p_1 and p_2 (represented in 2D by thick solid black lines) are of the same area, p_1 subtends a greater solid angle at the source S as it is closer to the source than p_2 ($d_1 < d_2$). Hence, the beam B_1 is wider than B_2 and carries more energy. Polygon p_3, which is of the same area as p_4, subtends a greater angle at the source due to the orientation of p_4. Hence, the beam B_3 is wider than B_4 and carries more energy in comparison. Among beams B_1 and B_3, we see that B_1 is wider since the surface area of p_1 is greater than that of p_3. Ideally, these beams must be given priorities based on decreasing order of energies they carry. Hence, we assign beam-width factors to the four beams in the order $F_{bw}(1) > F_{bw}(2) > F_{bw}(3) > F_{bw}(4)$.

The solid-angle subtended by a surface at the image-source is computed in spherical coordinates as

$$\Omega = \iint_{S_{poly}} \sin\theta d\theta d\phi. \tag{3.1}$$

For an irregular polygon, the surface limits are irregular and computation of the surface integral is not straightforward. The integral can only be computed by Monte-Carlo integration, which is a computationally expensive process. For the purpose of prioritization of beams, an approximation of the solid angle subtended should be sufficient. Hence, we use an analytical method, which is cheaper than Monte-Carlo integration, to compute the solid-angle subtended by an equivalent disk.

First, we find the projection of the polygon on a plane perpendicular to a vector from the image-source to the polygon's centroid, containing the closest vertex of the polygon (see p_4' in Fig. 3.1). Given the area of the reflecting polygon A_{poly}, its normal vector $\hat{n}_{poly}$ and the unit vector from the image-source to its centroid $\hat{v}_{s-poly}$, the area of its projection A_{proj} is given by,

$$A_{proj} = A_{poly}(\hat{n}_{poly}\hat{v}_{s-poly}). \tag{3.2}$$

We define the equivalent disk of a reflecting polygon as a disk of the same area as its projection A_{proj} to compute its solid-angle. The radius of the sphere that passes through

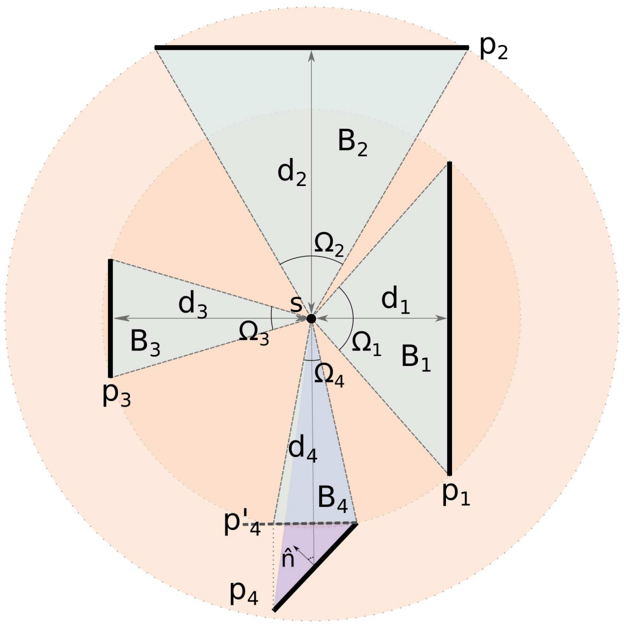


Fig. 3.1: Beams and their respective beam-width factors. The thick, solid black lines represent reflecting polygons p_{1-4}, the thick dashed line denotes projection p'_4 of the polygon p_4 on a plane tangential to the image-source, the blue triangles represent beams B_{1-4} and the peach coloured disks represent spheres passing through the vertices of the polygons. The beams B_1, B_2, B_3 and B_4 subtend solid angles Ω_1, Ω_2, Ω_3 and Ω_4 respectively and their beam-width factors are in the order $F_{bw}(1) > F_{bw}(2) > F_{bw}(3) > F_{bw}(4)$. The transparent violet coloured beam behind B_4 is the actual beam that is incident on the polygon p_4, for which the beam-width Ω_4 is an over-estimation.

the circumference of the equivalent disk is given by[1],

$$R_{sphere} = \sqrt{r_{eq}^2 + d^2}, \tag{3.3}$$

where r_{eq} is the radius of the equivalent disk, given by $r_{eq} = \sqrt{A_{proj}/\pi}$, and d is the distance of the centre of the disk from the image-source.

[1]Reference: https://physics.stackexchange.com/questions/331883/solid-angle-subtended-by-a-circle

The area of the sphere that the disk projects onto is given by,

$$\begin{aligned}
A &= \int_0^{2\pi} d\phi \int_0^{\tan^{-1}(\frac{r_{eq}}{d})} d\theta R_{sphere}^2 \sin\theta, \\
&= 2\pi(r_{eq}^2 + d^2)\left(-\cos\theta\right)_{\theta=0}^{\tan^{-1}(\frac{r_{eq}}{d})}, \\
&= 2\pi(r_{eq}^2 + d^2)\left(1 - \cos\left(\tan^{-1}(\frac{r_{eq}}{d})\right)\right), \\
&= 2\pi(r_{eq}^2 + d^2)\left(1 - \frac{d}{\sqrt{r_{eq}^2 + d^2}}\right). \qquad (3.4)
\end{aligned}$$

Hence, the solid angle subtended by the equivalent disk, and hence the width of the beam is given by,

$$\Omega = 2\pi\left(1 - \frac{d}{\sqrt{r_{eq}^2 + d^2}}\right). \qquad (3.5)$$

Computing solid-angle as explained above leads to an over-estimation of solid-angles subtended by surfaces whose normal vectors do not point towards the image-source. This is shown in Fig. 3.1, where the violet coloured beam incident on the polygon p_4 is the true beam and the computed beam B_4 is an over-estimation. However, the over-estimation never exceeds the solid angle of the disk if its normal was pointing at the image-source.

Given that the solid-angle of a whole sphere is $4\pi\ \ steradian$, the energy carried by a beam b is given by,

$$E_b = E_{emitted}\frac{\Omega_b}{4\pi}, \qquad (3.6)$$

where $E_{emitted}$ is the energy emitted by the sound source and Ω_b is the solid-angle of the beam. We define the band-width factor F_{bw} for the beam b as the energy carried by the beam when the energy emitted by the source is 1-unit, which is the ratio of the solid-angle of the beam to the solid-angle of a whole sphere, given by,

$$F_{bw}(b) = \frac{\Omega_b}{4\pi}. \qquad (3.7)$$

This factor aides in tracing beams deeper into the beam tree in directions where there is least amount of loss due to beam splitting, hence increasing the probability of early detection of valid higher-order reflection paths.

3.2.2 Air-absorption Factor

There is a possibility that two beams have equal beam-widths, carrying equal energies, but have travelled different distances. One such case is shown in Fig. 3.2. The paths resulting from the two reflecting polygons after path validation (see Sec. 2.3.3) will have different energies due to air-absorption and spherical spreading [42]. In the context of beams, using $1/r$ loss due to spherical spreading is not physically coherent (assuming the source emits spherical wavefronts) since beams diverge radially and the integral of sound pressure on the radiating face of the beam at any given distance from the source is equal to the integral of the sound pressure at any other distance (considering there is no loss due to absorption), i.e., there is no net loss due to spherical spreading within a beam. However, air-absorption can differentiate between the two beams by quantifying the amount of loss in energy during transmission. This would help in giving more priority to a beam which has travelled a shorter distance over a beam which has travelled a longer distance before being incident on the reflecting surface, in a physically consistent way.

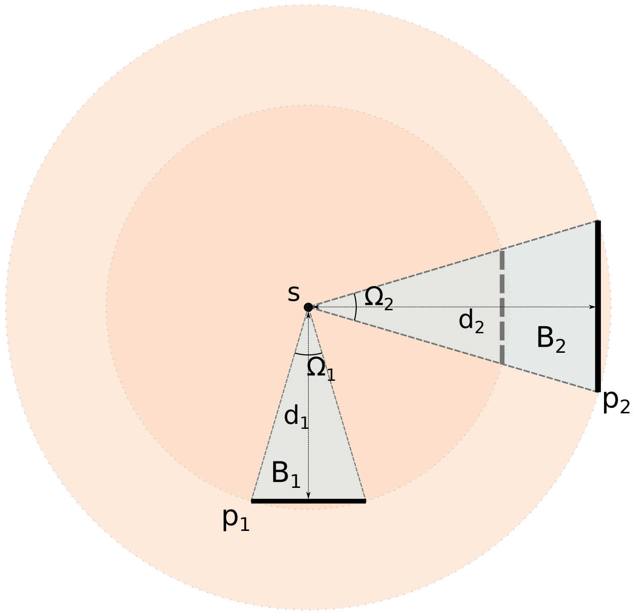


Fig. 3.2: Beams and their respective air-absorption factors. The thick, solid black lines represent reflecting polygons p_{1-2}, the thick dashed line represents the area of p_1 for comparison with area of p_2, the blue triangles represent beam B_{1-2} and the peach coloured disks represent spheres passing through the vertices of the polygons. The beams B_1 and B_2 subtend equal solid angles Ω_1 and Ω_2 at the source, respectively, and their beam-widths are the same. However, since $d_1 < d_2$, their air-absorption factors $F_{air}(1) > F_{air}(2)$.

Attenuation due to air-absorption depends on the frequency of sound, temperature and relative humidity of the atmosphere [15]. It is caused by shear viscosity, thermal conductivity and molecular relaxation due to vibrational, rotational and translational energy of oxygen, nitrogen and water vapour in air. The formula for computing the air-absorption coefficient for pure tones at a given temperature and relative humidity is given in [43]. However, for assigning priorities to beams based on air-absorption, a single coefficient representing the entire frequency range would simplify the computations greatly.

A single air-absorption coefficient can be computed for a band of frequencies by computing the area under the curve in Fig. 2.2 normalized over the bandwidth. In order to make it perceptually meaningful, we split computations over 10 octave bands in the audible frequency range and use the average of these air-absorption coefficients as the single value representing the entire spectrum. This way, since the individual coefficients are normalized over their respective bandwidths, we would essentially be giving more priority to low and mid-frequencies than to the higher frequencies where human hearing is less sensitive, hence retaining perceptual relevance of the air-absorption coefficient in a vague sense. The final air-absorption coefficient may, however, be made more perceptually coherent by using a weighted-average approach where the air-absorption coefficients of individual frequency bands are scaled by weights computed using an accurate perceptual model.

The overall air-absorption factor is given by,

$$\alpha_{air} = \frac{\sum_{i=1}^{10} \alpha_i}{10}, \tag{3.8}$$

where i is the index of the octave band. The octave bands are computed between 31.5 Hz and 20 kHz. The air-absorption coefficient α_{air} is pre-computed and the value is used for the entire duration of the simulation.

We define the air-absorption factor for a beam as the amount of attenuation of energy within the beam for the distance it has traveled. This distance is computed as the distance between the centroid of the reflecting polygon and image-source (d), which essentially is the average of the distance of each point on the polygon from the image-source. The air-absorption factor F_{air} of a beam b is given by,

$$F_{air}(b) = \left(\frac{1}{\alpha_{air}}\right)^{d}. \tag{3.9}$$

3.2.3 Material-Absorption Factor

The reflecting surfaces that make up the room geometry are made of materials with certain characteristic acoustic properties. They can absorb some of the incident sound, transmit some of it through the surface and reflect the rest of the energy. In this thesis, we consider only absorption and reflection of sound by the surfaces and ignore the transmission properties of materials. However, for the purpose of prioritization of beams, transmitted energy can be easily accounted for by considering it as a part of the absorbed energy.

The EVERTims software package [13,14] uses a database of common construction materials with their-absorption coefficients given for 10 octave bands between 31.5 Hz and 20 kHz. Table 3.1 shows the material-absorption coefficients for some of the materials listed in the database. In this thesis, we use the same database to account for material-absorption during the beam tracing step.

Table 3.1: Absorption coefficients of common construction materials (from the EVERTims Software package [14]).

	Octave Band Number									
Material	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Absorber	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Mirror	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Brick	0.030	0.030	0.030	0.030	0.040	0.050	0.070	0.090	0.150	0.150
Woodfloor	0.150	0.150	0.110	0.100	0.070	0.060	0.070	0.230	0.051	0.050
Concrete	0.360	0.360	0.440	0.310	0.290	0.390	0.250	0.230	0.051	0.050
Windowglass	0.350	0.350	0.250	0.180	0.120	0.070	0.040	0.230	0.051	0.050
Plywoodpanel	0.280	0.280	0.220	0.170	0.090	0.100	0.110	0.230	0.051	0.050
Halfdiffuser	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.050	0.050
Diffuser	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.050	0.050
Carpet	0.020	0.020	0.060	0.140	0.370	0.600	0.650	0.700	0.850	0.850
Plaster	0.013	0.013	0.015	0.020	0.030	0.040	0.050	0.070	0.050	0.050
Acoustile	0.260	0.260	0.570	0.630	0.960	0.440	0.560	0.700	0.850	0.850
Water	0.008	0.008	0.008	0.013	0.015	0.020	0.025	0.030	0.055	0.050
Marble	0.010	0.010	0.010	0.010	0.010	0.020	0.020	0.030	0.050	0.050

The energy absorbed by a material m in the octave band i is given by,

$$E^i_{abs} = E^i_{inc}\alpha^i_m, \tag{3.10}$$

where E^i_{inc} is the energy in octave band i incident on the surface and α^i_m is the absorption coefficient of the material m in octave band i.

The energy reflected by the surface in the octave band i is,

$$\begin{aligned} E^i_{refl} &= E^i_{inc} - E^i_{abs}, \\ &= E^i_{inc}\left(1 - \alpha^i_m,\right) \\ &= E^i_{inc}\gamma^i_m, \end{aligned} \tag{3.11}$$

where γ^i_m is the reflection coefficient of the material m in the octave band i.

For simplicity of computation, we consider the average of the reflection coefficients of all octave bands as the single reflection coefficient γ_m of the material (like in Sec. 3.2.2). Hence, the energy reflected by the surface of material m is,

$$E_{refl} = E_{inc}\gamma_m, \tag{3.12}$$

where $\gamma_m = \frac{\sum_{i=1}^{10} \gamma^i_m}{10}$. The energy of a beam reflected by a surface after N successive reflections, not considering air-absorption, is given by,

$$\begin{aligned} E_{b_{refl}} &= \left(\left(E_b\gamma_{m_1}\right)\gamma_{m_2}\right)...\gamma_{m_N}, \\ &= E_b \prod_{m \in M} \gamma_m, \end{aligned} \tag{3.13}$$

where $m_1, m_2, ...m_N \in M$ and M is the set of all materials that the reflecting surfaces are made of and E_b is the energy of the beam emitted by the source which has undergone N successive reflections.

We compute the priority factor due to material-absorption of the energy carried by a beam as the overall reflection coefficient of all the reflecting surfaces along the same branch up to the root of the beam tree. The material-absorption factor F_{mat} of a beam tree node b is given by,

$$F_{mat}(b) = \prod_{m \in M} \gamma_m. \tag{3.14}$$

This factor accounts for the total energy leaving a reflecting surface after reflection.

3.3 Interpretation of the Priority Factors

The three prioritization factors, namely the beam-width factor, air-absorption factor and material-absorption factor, together signify the energy carried by a beam. The beam-width factor signifies total energy emitted by the image-source within the width of the beam, the air-absorption factor accounts for the loss of energy during transmission of the beam in air and the material-absorption factor accounts for the loss of energy due to material-absorption. The final energy obtained after applying all the factors is the energy of the beam reflected by the surface.

$$E_{refl}(b) = E_{emitted} F_{bw}(b) F_{air}(b) F_{mat}(b), \tag{3.15}$$

where $E_{refl}(b)$ is the energy of the reflected beam and $E_{emitted}$ is the energy emitted by the source. The overall priority score of a beam is then given by,

$$F(b) = F_{bw}(b) F_{air}(b) F_{mat}(b). \tag{3.16}$$

Computing the priority factors in dB-scale would reduce the number of multiplications and exponential operations to perform by a large number, hence reducing the computational cost. In the dB-scale, the priority factor is given by,

$$F(b)_{dB} = F_{bw}(b)_{dB} + F_{air}(b)_{dB} + F_{mat}(b)_{dB}. \tag{3.17}$$

In the dB-scale, the exponentiation in Eq. 3.9 becomes a simple multiplication and the product term in Eq. 3.14 becomes a summation. The only complex operation is the conversion of beam-width factor to dB-scale. Since the beam-width factor, as is, involves for-loops for determining the centroid and area of the polygon, several square-root operations for computing the distance between points and computation of the solid-angle, apart from many floating-point multiplications and divisions, it is the most computationally demanding and dictates the cost of computation of the total beam priorities. However, the main advantage of using the dB-scale is that floating-point errors are avoided when the priority factors take very small values. The computation of the air-absorption factor in dB-scale only adds one floating-point multiplication and one floating-point addition, and the computation of the material-absorption factor in dB-scale for a single beam only adds two floating-point addition operations, hence are insignificant.

As beams are traced and the beam tree gets deeper, the amount of energy that they carry decreases due to beam splitting (from partial incidence on surfaces), greater air-absorption during transmission and greater loss of energy due to absorption by the reflecting surfaces. The beam-width helps in prioritizing beams that are wider, which means they carry more of the emitted energy, amount of air-absorption helps in prioritizing beams that have traveled shorter distances, and finally, amount of material-absorption helps in prioritizing beams that have undergone fewer reflections and those which are reflected by materials that absorb less energy.

As we go higher in reflection order, the beams get narrower and narrower due to splitting and larger distances between the reflecting polygons and the image-sources, hence having a lower chance of containing valid paths. Similarly, as the beams are traced through longer distances and more reflections, they undergo more and more attenuation, hence yielding paths of lower energy and perceptual significance. Together, the three prioritization factors, and hence the total energy carried by a beam, tell us about the likelihood of a perceptually significant valid path (the perceptual significance comes from the average of the octave-band-wise air-absorption and material-absorption coefficients) from the source to the listener existing within the extent of the beam. Using this energy estimate to govern the choice of the next beam to trace at any given iteration of the beam tracing step will result in a beam tree where perceptually significant reflection paths from the source to the listener occur earlier, hence having to trace fewer beams overall, compared to depth-first and breadth-first beam tracing.

To illustrate this, let us take an example of a room which is very long and narrow, like a narrow hallway as shown in Fig. 3.3. When the source and the listener are very close to each other, the higher order specular reflections that reflect off the side walls, floor and the ceiling arrive at the listener before some lower order specular reflections off the front and the back walls, which, in this case, are of less consequence for the perception of the immediate surroundings of the listener. This is where the prioritized beam tracer shines. Unlike the depth-first beam tracer that traces beams based on the branch they belong to or the breadth-first beam tracer that traces beams hierarchically (traces order-by-order from the root to the leaf), the prioritized beam tracer gives a fair chance to all untraced beams, choosing the beam that contains the highest energy for further beam tracing, irrespective of branch or order. This makes the beam tracer more efficient in environments where the immediate surroundings of the source and the listener contribute significantly to the

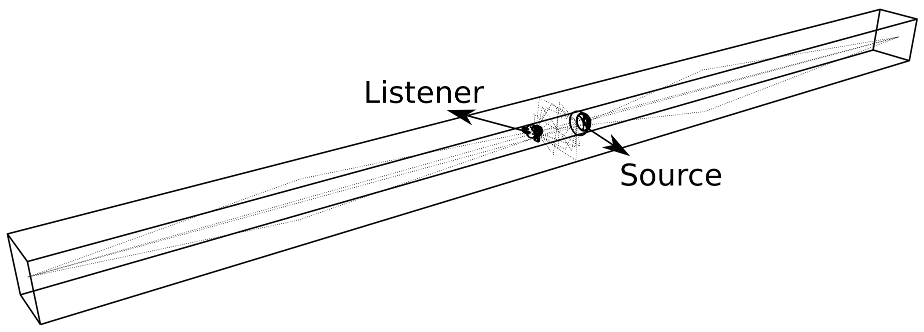


Fig. 3.3: Reflection paths in a narrow corridor (represented by light grey lines from the source to the listener). Here, we can see that the reflection paths from the side-walls, floor and ceiling arrive at the listener earlier than the reflections from the walls on the opposite ends of the corridor. The prioritized beam tracer would give more importance to the reflections from the immediate surroundings of the source compared to the surfaces that are further away.

specular reflection energy, giving us the more significant higher-order reflections before less significant lower-order reflections.

The priority of a beam is essentially a likelihood estimate. The beam-width factor tells us about the likelihood of a beam containing a valid reflection path to the listener. A good explanation for this can be found by comparing a prioritized beam tracer that uses only the beam-width factor to a stochastic ray-tracer. From the perspective of the beam tracer, a small reflecting surface or a reflecting surface that appears to be small to an N^{th}-order image-source due to the distance between them results in a beam of low beam-width factor. From the ray-tracer's standpoint, the probability that a ray emitted by the source strikes a small reflecting surface or a reflecting surface after N reflections is small since only a fraction of the rays emitted manage to reach the surface. The beam-width factor in the case of the beam tracer and the probability of ray incidence in the case of the ray-tracer can be seen as being equivalent. Hence, prioritizing beams with higher beam-width factors is equivalent to prioritizing rays that have a greater probability of reaching the listener. The beam-width factor also helps in growing the beam tree deeper in certain directions due to the likelihood of finding valid reflection paths.

The other two priority factors account for the attenuation during the transmission of beams, hence guiding the beam tracing in directions where there is more likelihood of finding perceptually significant reflection paths. This can be compared to a stochastic ray-tracer in which rays are terminated on falling below certain energy threshold or shot down in a "Russian Roulette" fashion based on the probability of eventually reaching the listener [31]. However, the probability would only make sense if the amount of energy that is initially emitted by the source in that direction is known. Hence, the absorption based beam priority factors need to be used in combination with the beam-width factor for them to be physically meaningful, thereby signifying the likelihood of finding perceptually significant valid reflection paths. However, in the rest of the thesis, we will not be taking the route of probabilities, but rather stick to the energy interpretation of the beam priorities.

Even within the absorption based priority factors, the amount of influence they have on the overall results obtained by the prioritized beam tracer depends on the room model. The air-absorption priority factor would have a greater influence on the results in rooms of large dimensions and greater mean free paths, since air-absorption without spreading losses is rather insignificant for smaller travel distances (around -0.08 dB/m at 23^oC and 35% relative humidity). The material-absorption factor would have a greater influence in room models with more absorptive surfaces. Deviating from the physical coherence by weighting the priority factors could lead to greater improvements in the results of the prioritized beam tracer. However, in this thesis we focus on the analysis of physically coherent prioritization only.

3.4 Termination Condition

In the depth-first and the breadth-first beam tracers, a pre-defined maximum allowed order of reflections is used as the termination condition. Since the prioritized beam tracer does not explicitly consider the reflection order during beam tracing, the question of when to stop tracing beams arises. Using a maximum order of reflections as a termination condition would lead to an overall increased cost due to the added computation of beam energies (priority scores) and maintenance of additional data structures for storing untraced beams. Also, depending on the room geometry and the position of the source, some significant higher-order reflections may be missed if the beam tree is truncated at a low reflection order.

Since the prioritized beam tracer is expected to trace beams that have more significant valid reflection paths to the listener earlier, terminating the beam tracing process after tracing a specified number of beams may be sufficient. When compared with a full beam tree up to certain reflection order, the beam tree obtained using this technique may be incomplete and may result in missed paths. Nonetheless, since the missed paths may be insignificant compared to the identified ones, this technique would result in a gain in the overall computation time at the cost of a small loss in accuracy. As a consequence, the computation time becomes fixed and there would be little to no change in time taken for different room models. However, since the width of a beam tree depends on the complexity of the room geometry, i.e., a simple room with fewer polygons would result in a narrower beam tree compared to a complex room with a larger number of polygons, the number of beams to trace before termination for a required level of accuracy must be determined through offline simulations.

A more physically sensible way to truncate the beam tree would be to trace all beams carrying energies above a specified threshold and rejecting the rest. Imposing a minimum requirement on beam energies would ensure exhaustion of beams to trace, hence terminating the beam tracing process. This method, like the previously mentioned method, suffers from the drawback of missing paths. However, since fewer beams are traced, this technique could improve the computation time at the cost of a small loss in accuracy. Nonetheless, both techniques would greatly speed up the path validation step further since the size of the resulting beam trees would be smaller.

A fixed number of beams, however, can be used as a pausing condition in an iterative refinement framework for real-time simulations. Here, the beam tracing process is paused on tracing a specific number of beams for performing path validations, after which the beam tracing process may resume and again stop on tracing a predetermined number of beams. The solution provided by this beam tracer gets better with time for a stationery source since all untraced beams are saved and would eventually be traced. However, maintenance of a priority queue gets more and more expensive as more beams are stored, hence exponentially increasing the cost of computation with each iteration.

If a threshold as described above is used as a pausing condition, where beams that do not make a certain threshold are thrown away, one would have to compute new trees from scratch for each threshold. This problem can be seen even in the pseudo-breadth-first implementation of the beam tracer in the EVERTims software package, where the leaf nodes at any iteration of the beam tracing process are not kept track of for resuming the beam tracing process, hence forcing completely new beam trees to be built for each reflection order. A pure threshold based implementation would be worse than the pseudo-breadth-first implementation in an iterative refinement framework due to the additional costs of priority computations and beam maintenance. However, an adaptive threshold may be used to pause the beam tracing process, where untraced beams that do not make the threshold at a given time are not thrown away, but are stored for future tracing. This method, however, has the drawback of increasing cost of computation with every iteration as in tracing an incremental number of beams since even clearly neglectable beams are also saved.

A hybrid solution making use of a hard-threshold to reject insignificant beams whose priority is below the threshold along with either an adaptive threshold or an incremental number of beams for pausing the beam tracing process may be used to iteratively refine the obtained solution, thereby exploiting the advantages of both techniques. Table 3.2 provides a summary of the pros and cons of the terminating conditions discussed.

In the next chapter, we evaluate the performance of the prioritized beam tracer comparing it with the depth-first, breadth-first and the pseudo-breadth-first beam tracer used in the EVERTims software package (Sec. 2.4) with the various terminating criteria mentioned above and also comment on the suitability of the algorithm for use in real-time simulations.

Table 3.2: Comparison of termination conditions for prioritized beam tracing and their suitability for real-time simulation.

		Order of Reflections	Number of Beams	Priority Threshold	Hybrid: Adaptive Pausing with Fixed Minimum Priority
Completeness of beam tree	**Pros**	Complete beam tree. Finds all paths up to the given order.	Beams with more priority are traced first, hence result in more representative beam trees for the same tree size.	A more physical way to terminate beam tracing. Result in more representative beam trees for the same tree size.	Smooth convergence towards a complete beam tree when used with a upper limit on reflection order. Significant paths are identified in earlier iterations.
	Cons	Significant beams of higher order are not traced, hence could miss significant higher order paths.	May result in incomplete beam trees up to a given order. Could miss paths.	May result in incomplete beam trees up to a given order. Could miss paths.	Incomplete beam trees due to rejection of insignificant beams
Computational cost	**Pros**		Near constant computational cost. Cost can be tuned by varying the number of beams traced. Fewer beam tree nodes for path validation.	Fewer beam tree nodes to be validated. No insignificant nodes saved, hence lower cost of priority queue maintenance.	Computation costs spread across time. Lower cost of priority queue maintenance due to rejection of insignificant beams.
	Cons	Computation cost worse than depth-first and breadth-first beam tracers due to priority computations.	Increased queue maintenance cost due to insignificant beams in the queue. Needs offline simulations to determine the number of beams to trace for a given geometry.	Unpredictable computation costs. Needs offline threshold tuning.	Varying computation times between iterations.
Real-time?		Higher cost than EVERTims. slower for the same resulting paths.	Increased cost due to maintenance of priority queue. Number of beams traced may be tuned for real-time purpose.	May be used with high thresholds. Iterative refinement would be expensive due to redundant computations.	Highly suitable for real-time simulations with iterative refinement.

Chapter 4

Evaluation of the Prioritized Beam Tracer

In this chapter, we present the test platform, room models and configurations chosen for simulations and a thorough evaluation of the proposed prioritized beam tracer's performance compared to the previously presented beam tracing approaches. We also discuss the suitability of the physically-informed prioritized beam tracer in a real-time context and its tunability for applications with different requirements.

4.1 Evaluation Setup

4.1.1 Platform

The prioritized beam tracer is implemented by embedding the prioritization of the beams into *libevert*[1] , which is an open source implementation of the depth-first beam tracer in C++ by Laine et al. [1], available under the *GNU General Public License*, along with additional data structures for storing information about untraced beams. The choice of embedding the prioritization into an already available implementation was to facilitate quick prototyping. Since this thesis deals almost entirely with the beam tree construction phase of the beam tracing algorithm, the path validation segments of the original implementation that include optimizations mentioned in Sec. 2.3.3 are left untouched.

The choice of beams to trace in the prioritized beam tracing process depends on the energy/priority scores of the beams (computed in dB-scale as discussed in Chapter 3). When a new beam is constructed (due to a reflection), the beam-width factor and the air-absorption factor are computed explicitly for the beam and the material absorption factor

[1]The implementation can be found in https://users.aalto.fi/~laines9/publications/laine2009aa_code.zip.

is computed by accumulating the absorption coefficient of the reflecting polygon with the material-absorption factor of its parent beam. The air-absorption coefficient used in simulation for the computation of the air-absorption priority factor is pre-computed as the average of absorption coefficients for 10 octave bands (as explained in Sec. 3.2.2), at a room temperature of 23^oC and a relative humidity of 35%. The material absorption coefficients for different materials are also pre-computed from the database of coefficients given for 10 octave bands.

In order to select the beam with the highest priority score for continuation of the beam tree construction process, we store the new beams in a priority queue, which is a data structure that stores the node with the highest score at the top, hence readily giving us the best beam to trace. We use the *std::priority_queue* implementation of the priority queue available in the *C++ standard template library*, which is essentially a max-heap with constant-time look-up and O(log n) average and worst case insertion and deletion complexities. Once a beam is traced, the reflecting surfaces are added to the beam tree, the priorities of the reflected beams are computed and the beams are then pushed into the priority queue. The next beam chosen for tracing is the beam with the highest priority among the untraced beams, which is at the top of the priority queue.

For the comparative study presented in this chapter, two more beam tracers are implemented: a breadth-first beam tracer and a pseudo-breadth-first beam tracer like the one used in the EVERTims software package. The breadth-first beam tracer performs beam tracing by order of reflections, i.e., it traces all beam of one reflection order before moving on to the next reflection order. It is implemented by using a queue for storing new beams, which is a linear data structure that follows a first-in-first-out (FIFO) methodology, with linear insertion and deletion costs. When new beams are constructed on reflection, they are simply pushed to the back of the queue, and the next chosen beam for tracing is the beam in front of the queue. The pseudo-breadth-first beam tracer (referred to as the EVERTims beam tracer henceforth) performs beam tracing by reflection order, but builds new trees for each order of reflection in a depth-first manner. For example, the EVERTims beam tracer first builds a beam tree of reflection order 1, then builds a new beam tree for order 2, order 3, and so on. Since completely new beam trees are built for each order, there is no need of a data structure to store the leaf nodes of the beam tree of any given order. The performance of the prioritized beam tracer is compared against the performance of the original depth-first beam tracer by Laine et al. [1], the breadth-first beam tracer and the EVERTims beam tracer.

All simulations performed as part of this thesis were done on a computer with an Intel i7-7700K, 4.2 GHz 4-core processor and 32 GB of 3 GHz DD4 RAM. The simulations were performed such that a single thread was used on a single processor core for the entire simulation. The beam tracing results were written into .bin files to be parsed in Matlab for analysis.

4.1.2 Room Geometries and Occlusion Configurations

The beam tracers are tested in three different room geometries to evaluate their performance in environments with varying complexities made of different number of convex polygonal surfaces:

1. **Cube:** Simple geometry with 6 convex surfaces and no occlusions. (Fig.4.1)
2. **Simplified apartment:** An apartment model with four rooms made of 53 convex polygonal surfaces with thin walls (walls do not have thickness). (Fig.4.2)
3. **Complex apartment:** The same apartment model as the simplified apartment, but with walls having thickness (boxy structures made of up to 6 convex polygons) and tessellations, made of 315 convex polygons. (Fig.4.3)

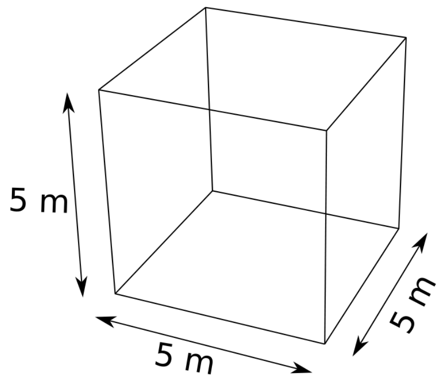


Fig. 4.1: Room Model: Cube made of 6 convex polygonal surfaces.

The two apartment models are based on the same floor-plan and are of the same dimensions. However, the complex apartment model is designed such that the complexity of the room geometry in terms of the number of polygons that it is made of is increased. The complex apartment has thick walls, i.e., each wall is defined by more than one polygon covering the outer surface, with unnecessary tessellations of the reflecting surfaces.

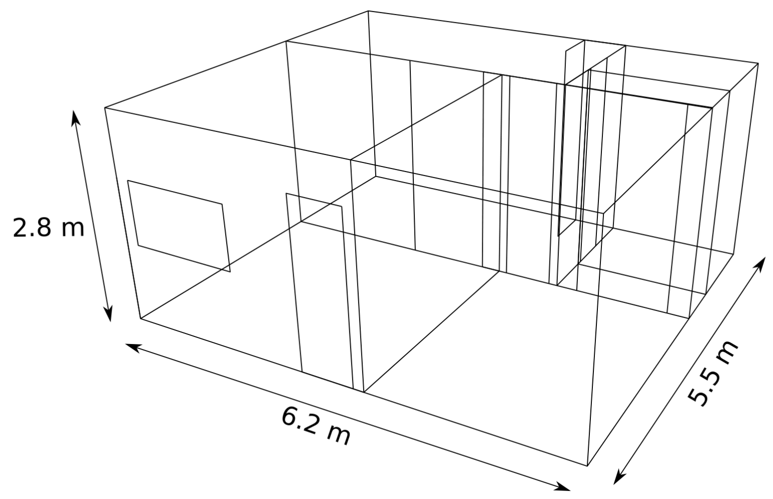


Fig. 4.2: Room Model: Simple apartment made of 53 convex polygonal surfaces. The walls in the apartment have zero-thickness, i.e., a single surface defines an entire wall.

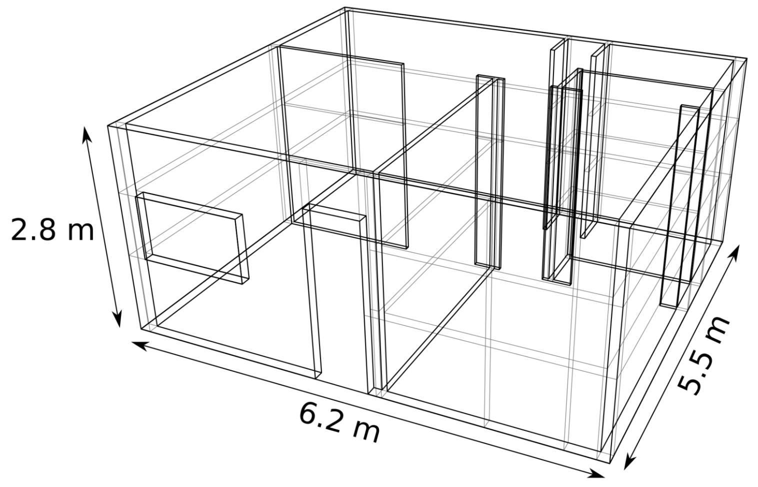


Fig. 4.3: Room Model: Complex apartment made of 315 convex polygonal surfaces. The apartment model is the same as in Fig.4.2, but with walls having finite thickness, i.e., they have a finite volume, made of up to six bounding surfaces.

Hence, it is a poorly constructed room model. On the other hand, the simplified apartment model is a very well constructed room model with minimum tessellations and thin walls, i.e., each wall is represented by a single polygon. Although the obtained specular reflection energies in the two rooms would be similar (not exactly the same due to difference in wall thicknesses), the beam trees constructed for the complex apartment model will be far wider with a far higher number of beams traced. The beams that are traced in the complex apartment model would also be narrower due to tessellations in the reflecting surfaces (shown as light lines in Fig. 4.3) giving rise to smaller reflection polygons. Hence, one may expect the beam-width factors, and hence the priorities of the beams, to be a lot less compared to the beams in the simplified model. This has certain implications that are discussed in the end of Sec. 4.3.

Surfaces in the cube model are made of randomly chosen materials from the database. The apartment models are made of *concrete* side walls, *plaster* ceilings and *wooden* floors for realism. Since we are comparing the beam tracers within the same models, the choice of materials does not adversely affect the overall comparison. However, if surfaces of lower absorption coefficients are chosen, the material absorption factor would be of less consequence in the prioritization of beams.

In order to compare the performance of the different beam tracers against each other while staying within memory limits of the computer, we limit the maximum reflection order at 10 for the cubical room and 6 for the two apartment models. In the case of the cube model, there is no occlusion for any source and listener positions, i.e., the room is perfectly convex. However, in the apartment models, there may be light to heavy occlusion depending on the locations of the source and the listener. Hence, we test the apartment model in three occlusion configurations:

1. **Configuration 1:** Source and listener in the same room with no occlusion. In this configuration, paths of all orders of reflection, from low to high-orders, are guaranteed due to line of sight between the source and the listener and all walls around them reflecting sound back into the room, i.e., the immediate surrounding of the source and the listener is nearly convex.

2. **Configuration 2:** Source and listener in adjacent rooms with light occlusion. In this configuration, direct path between the source and the listener cannot be guaranteed, but mid- to high-order reflections would exist.

3. **Configuration 3:** Source and listener in disconnected rooms with heavy occlusion by the separating walls. In this configuration, only the high-order reflection paths are expected to exist between the source and the listener positions.

When there is occlusion between the source and the listener, sound from the source must either bend around the obstacle or undergo reflections by surrounding surfaces in order to reach the listener. Since the beam tracer can only model specular reflections, only the higher-order reflections reach the listener when there is an occlusion. In order to identify these higher-order reflections, the beam tracer must trace deeper into the tree. Since the prioritization of beams does not consider the order of reflection and traces beams only based on likelihood of a significant path detection, the beam tracer may be able to perform well even in occluded conditions. Hence, we verify the performance of the proposed beam tracer in the three mentioned levels of occlusion. Since there are 4 rooms in the apartment models, there can be 4 different pairs of room locations of the source and the listener in configuration 1 (both in the living room, kitchen, bedroom or bathroom), 3 different pairs of rooms in configuration 2 (living-kitchen, kitchen-bathroom, kitchen-bedroom) and 2 different pairs of rooms in configuration 3 (living-bedroom, bedroom-bathroom). We perform 30 trials with randomly chosen source and listener positions in each of these pairs of rooms.

4.2 Evaluation Metrics

In this thesis, we evaluate the beam tracers' performances primarily on the basis of how they identify specular reflection paths. Every reflection path identified by a beam tracer contributes a certain amount of energy to the total specular reflection energy reaching the listener. The energy of a reflection path (also referred to plainly as path energy) depends on its length and the amount of energy absorbed by the surfaces it is reflected by. We compute the energy E_p of a reflection path p as the squared sum of samples of the impulse response corresponding to the path, which includes spherical spreading loss [42], frequency-dependent air-absorption and loss due to absorption by reflecting surface, as follows:

$$E_p = \sum_{n=0}^{N-1} h_p[n]^2, \tag{4.1}$$

where $h_p[n]$ is the impulse response of the reflection path whose length in samples is N and is computed as,

$$h_p[n] = \frac{1}{r_p}\left(h_p^{air}[n] * h_p^{mat}[n]\right), \tag{4.2}$$

where r_p is the length of the reflection path, $h_p^{air}[n]$ is the impulse response of a filter approximating air-absorption for the reflection path and $h_p^{mat}[n]$ is impulse response of a filter approximating material-absorption for the reflection path.

The air-absorption filter is computed as given in [43] and the material-absorption filter is computed for each reflection path as the minimum phase response [44] of the weighted sum of octave band filters, with each filter weighted by the total absorption coefficient of all reflecting surfaces in that octave band.

The total specular reflection energy reaching a listener is the sum of energies of all specular reflection paths identified in the complete beam tree up to the predetermined order of reflections, given by,

$$E_{total} = \sum_{\forall p \in P} E_p, \tag{4.3}$$

where P is the set of all valid reflection paths identified by the beam tracer.

From our understanding of functioning of the different beam tracing algorithms used in the simulations, we can make some predictions on the results obtained from them. In the depth-first beam tracer, we expect valid paths to occur in a random fashion in the beam tree obtained, i.e., there is no organization of paths in terms of energy or where they occur in the beam tree since it traces beams branch-wise. In the breadth-first beam tracer, we expect the paths to be ascending in reflection order, distributed across the beam tree, which means there would be a generally decreasing trend in the energy of paths. In the EVERTims beam tracer, since the beam tracing is done in a pseudo-breadth-first manner, i.e., the results obtained are in breadth-first manner but the beam tracing itself happens in a depth-first manner for each order of reflection, we expect the occurrence of paths to be similar to the breadth-first beam tracer, but only spread out over a larger number of beams due to redundant retracing of beams (explained in Sec. 2.4). Finally, in the prioritized beam tracer, we expect the paths to occur in a mostly descending order of energy but packed towards the beginning of the beam tree, i.e., valid reflection paths are expected to be identified in fewer beam traces since beams are traced based on their priorities that come from physically-informed estimates of the energy that they carry.

Since the more significant reflection paths occur earlier in the beam tree, the plot of path energies vs. their beam tree node index would display a decreasing behaviour. Hence, the integral/cumulative sum of the path energies, when plotted against the number of beams traced before identifying the valid reflection paths, can be expected to display a steep increase in the beginning and gradually approach the total specular reflection energy as more and more beams are traced. This would somewhat resemble the cumulative distribution function of an exponential distribution. Since the beam tracing algorithms considered in this thesis are listener-dynamic and have no knowledge of the listener position at the time of beam tracing, the rate at which the cumulative path energy rises can vary depending on the listener position. However, in general we expect the prioritized beam tracer to show the quickest early rise in the specular reflection energy due to the judicious tracing of beams, followed by the breadth-first beam tracer and then the EVERTims beam tracer.

Simulation Cost. In order to evaluate the beam tracers while accounting for how efficiently they trace beams, we define a metric called the *Simulation Cost*, which is the number of beams that need to be traced in order to identify at least 90% of the total specular reflection energy E_{total}. Consider a single trial in which a total of M reflection paths are identified between the source and the listener. Let **B** $= \{B_1, B_2, ..., B_i, ...B_M\}$ be an ordered set of the number of beams traced before finding the corresponding reflection paths $\{p_1, p_2, ..., p_i, ...p_M\}$. The simulation cost for this trial is defined as the first occurrence of B_m where $m \in \{1, 2, ...M\}$ such that

$$\frac{\mathcal{E}_p^m}{E_{total}} \geq 0.9, \tag{4.4}$$

where

$$\mathcal{E}_p^m = \sum_{i=1}^{m} E_{p_i} \tag{4.5}$$

is the cumulative sum of path energies up to path p_m. This metric would help in characterizing the beam tracers by the distribution of paths in their respective beam trees according to the path energies. The lower the simulation cost, greater the compactness of paths in the beam tree and greater the efficiency of beam tracing.

Simulation Time. The simulation cost only talks about the compactness of the distribution of paths in the beam tree and not about how quickly in time the cumulative path energy rises in each of the beam tracers. The prioritized beam tracer proposed in this thesis has an added cost of priority computation, which is not there in the other beam tracers. Hence, in order to compare their timing performances, we define a metric called the *Simulation Time*, which is the time taken by the beam tracer to identify at least 90% of the total specular reflection energy. Consider a single trial in which a total of M reflection paths are identified between the source and the listener. Let **T** $= \{T_1, T_2, ..., T_i, ...T_M\}$ be an ordered set of elapsed times, where T_i is the time elapsed since the beginning of beam tracing until the path p_i is identified, without accounting for path validation. As for the case of simulation cost, the simulation time for the trial is defined as the first time T_m where $m \in \{1, 2, ...M\}$ such that

$$\frac{\mathscr{E}_p^m}{E_{total}} \geq 0.9, \tag{4.6}$$

where $\mathscr{E}_p^m$ is defined as in Eq. 4.5.

For both the simulation cost and simulation time, we choose the 90% mark as the point of comparison of the beam tracers since it signifies a significant portion of the total specular reflection energy. However, we will see in the following sections that choosing a different percentage of the total specular reflection energy as the point of comparison of the beam tracers only changes the amount by which the prioritized beam tracer performs better than the other beam tracers, but does not take away the fact that the prioritized beam tracer performs the best.

In the following sections, we first compare the beam tracers based on the distribution and organization of valid reflection paths by their energies in the room geometries and occlusion configurations mentioned in Sec. 4.1.2. As discussed in Sec. 3.4, the prioritized beam tracer gives us the capability to truncate the beam tree while maximizing accuracy since beams are traced based on physically-informed estimates of energy that they carry. Hence, we evaluate the use of the different termination and pausing conditions mentioned in Table 3.2. We then compare the beam tracers based on their timing performances in the mentioned room geometries and occlusion configurations. We also present a brief account on the influence of the prioritization factors on the average simulation costs and the simulation times of the beam tracers, along with a commentary on the tunability achieved due to judicious tracing of beams.

4.3 Distribution of Valid Paths in the Beam Tree

First, we perform simulations with the only termination condition being a maximum reflection order of 10 in the cube model and 6 in the apartment models, hence building complete beam trees up to the specified orders for the comparison of the distribution of paths in the beam tree. We mark a beam tree node as containing a valid path from the source to the listener if it passes the path validation test (see Sec. 2.3.3).

4.3.1 Performance in an Individual Trial

In order to get an idea on how the reflection paths are distributed in the beam trees constructed by the beam tracers, let us look at Figs. 4.4 - 4.6 showing examples of stem plots of the energy of paths plotted against their beam tree node indices for the different room models and occlusion configurations in a single randomly chosen trial.

In the stem plots shown in Fig. 4.4 for the cube model, we see that the depth-first beam tracer, breadth-first beam tracer and the prioritized beam tracers take almost equal number of beams to recognize all valid reflection paths. This is due to the fact that the room model is perfectly convex and all the beams reflect back into the room. However, the depth-first beam tracer has no organization in terms of path energies that the breadth-first and the prioritized beam tracers display. Both breadth-first and the prioritized beam tracers have, in general, paths occurring in decreasing order of their energies. In the breadth-first beam tracer, we can anticipate this kind of behaviour due to the fact that higher-order reflections travel further distances and are subject to greater attenuation. In the prioritized beam tracer, we expect this kind of behaviour since we prioritize beams that show greater likelihood of identifying high energy paths. The EVERTims beam tracer retains the general decreasing order of path energies as seen in the breath-first beam tracer, but takes more than twice as many beams to identify the paths, which is a consequence of the iterative order refinement technique employed. Hence, the breadth-first and prioritized beam tracers perform the best in very simple and convex geometries.

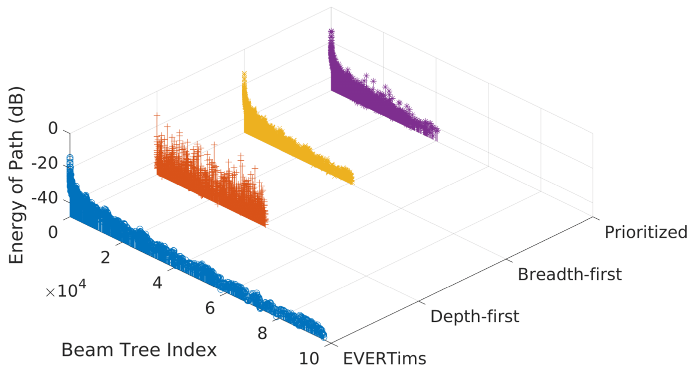


Fig. 4.4: Stem plot of path energy vs. beam tree node index - cube model.

In the stem plots shown in Fig. 4.5 for the simplified apartment model, we see that the depth-first beam tracer shows the same behaviour as in the cube model, where there is no organization seen in terms of path energy and the EVERTims beam tracer takes the most number of beams to recognize all the paths. In occlusion configurations 1 and 2, the breadth-first beam tracer clearly shows a decreasing trend in the energy of paths as expected, however, has large gaps in its beam tree where there are no paths at all. The prioritized beam tracer, apart from displaying a decreasing order in path energies, also shows compactness in its beam tree without significant gaps. This shows that tracing beams based on the likelihood of having valid reflection paths indeed helps in identifying reflection paths earlier in the beam tracing process. Hence the prioritized beam tracer performs the best with most of its significant paths occurring earlier in the beam tree compared to the other algorithms. Because of the compactness of the distribution of paths, the prioritized beam tracer takes fewer beams to identify all reflection paths. Even in configuration 3 where there is heavy occlusion, the valid reflection paths occur earlier in the beam tree of the prioritized beam tracer, whereas they are scattered across the beam trees of the other beam tracers.

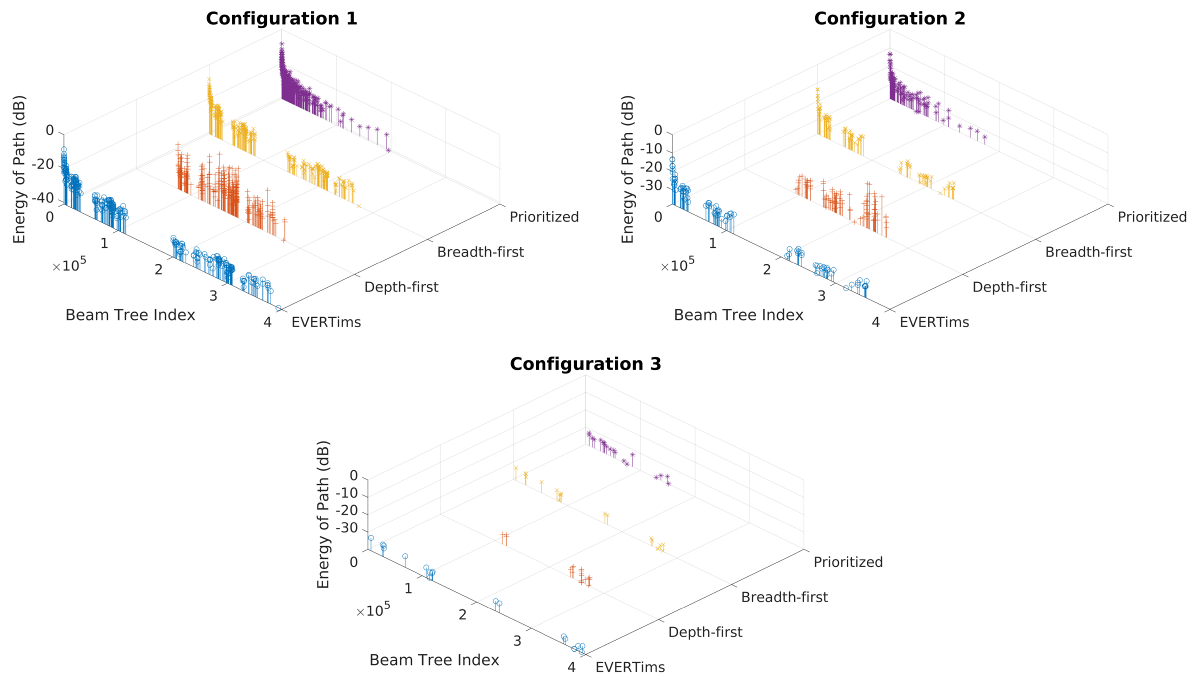


Fig. 4.5: Stem plot of path energy vs. beam tree node index for the simplified apartment model.

In the complex apartment case shown in Fig. 4.6, the beam tracers perform in a similar fashion as in the simplified apartment model. The prioritized beam tracer shows greater compactness in the distribution of paths than the other algorithms in all occlusion configurations. On careful observation, we can see that the prioritized beam tracer takes a smaller percentage of beams as the breadth-first beam tracer for finding all paths than in the simplified apartment model, i.e., there is more compactness in the occurrence of paths. This hints at the prioritized beam tracer providing greater benefits in more complex models.

4.3.2 Performance Across Trials

In order to verify the observations made in the stem plots for a general case and not just in an individual trial, we plot 2-D histograms of paths from all trials in a given room model and occlusion configuration, based on their energies and where in the beam trees they occur. Figures 4.7 - 4.9 show the 2-D histograms for the room models and occlusion configurations.

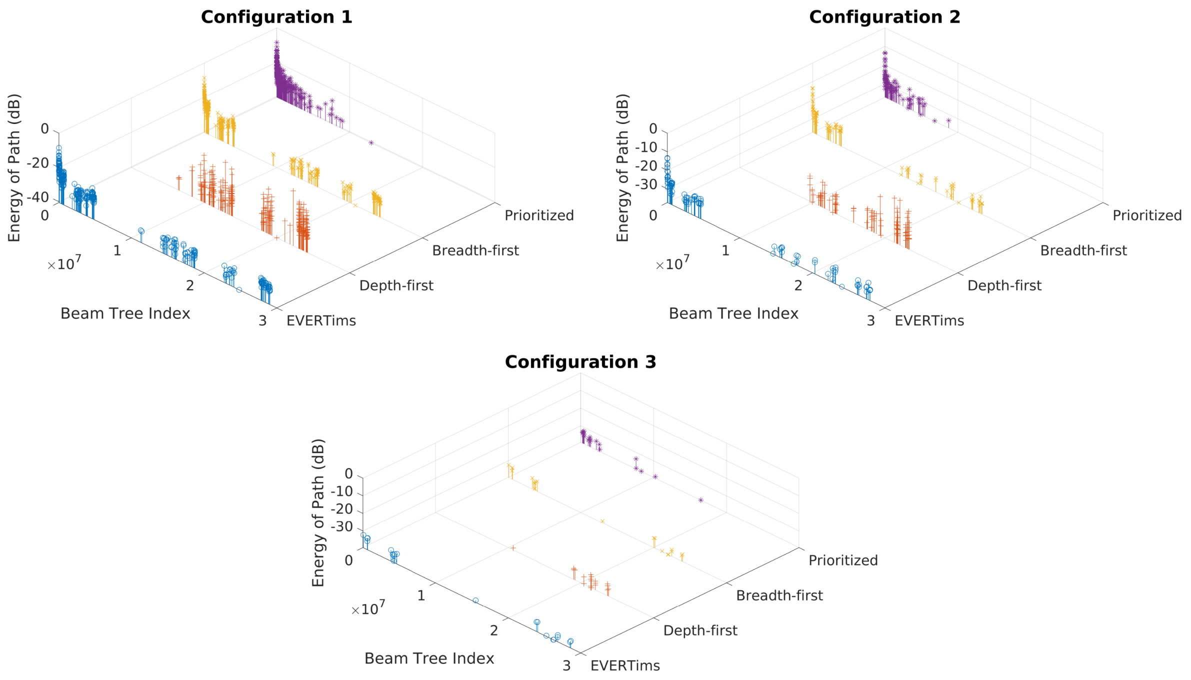


Fig. 4.6: Stem plot of path energy vs. beam tree node index for the complex apartment model.

We see from Fig. 4.7 showing histograms for the cube model that the breadth-first and the prioritized beam tracers take almost identical number of beams to identify paths in general, which was what we observed in the stem plot in Fig. 4.4. However, the histogram of the prioritized beam tracer displays a smoother distribution in energy of paths and beam tree index compared to the step-like distribution of the breadth-first beam tracer. This hints at a slightly better performance of the prioritized beam tracer over the breadth-first beam tracer. The histogram of the EVERTims beam tracer is similar to that of the breadth-first beam tracer, only stretched out further as expected.

In Fig. 4.8 showing the histograms for the simplified apartment model, we see that, in configuration 1, where the source and the listener are in the same room, the prioritized beam tracer shows more compactness, detecting paths earlier compared to the breadth-first beam tracer (takes roughly half as many beams as the breadth-first beam tracer), while displaying a general decreasing order of path energies in the beam tree. In configuration 2, where there is light occlusion between the source and the listener, there are gaps seen in the histogram of the breadth-first beam tracer, which shows that there are significant chunks of nodes in the beam tree where there are no valid reflections at all (as observed in the stem plots in Fig. 4.5). However, the compactness of the histogram for the prioritized beam tracer is retained, which confirms that the prioritized beam tracer traces beams in directions where there is a greater likelihood of significant valid paths. In configuration 3, where the source and the listener are heavily occluded, the paths are scattered across the beam tree in all the cases, however, they are spread over a smaller region towards the beginning of the beam tree in the prioritized beam tracer compared to the

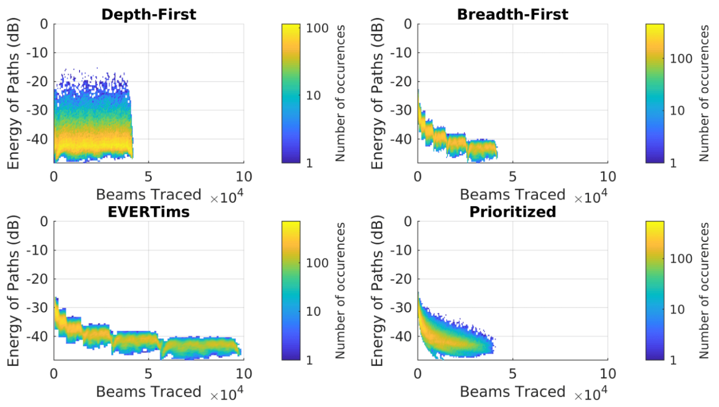


Fig. 4.7: Histograms of path energy vs. beam tree node index - cube model.

other beam tracers. Hence, we see that the prioritized beam tracer behaves as expected from the stem plots, displaying the best compactness in the distribution of paths in the beam tree ordered by energy. This proves that the prioritized beam tracer reaches lower in the beam tree by tracing higher-order reflection beams of higher priorities, therefore, detecting valid reflection paths earlier compared to the other methods regardless of the level of occlusion.

In Fig. 4.9, showing the histograms for the complex apartment model, we see that the beam tracers behave in a similar fashion as in the simplified apartment model. However, looking close, we see that the histograms of the prioritized beam tracer in all configurations of occlusion, show smaller spread than the breadth-first beam tracer in the complex apartment in contrast to the simplified apartment in respective occlusion configurations (for example, the prioritized beam tracer appears to take roughly 1/3rd the number of beams compared to the breadth-first beam tracer in the complex room model as opposed to half in the simplified room model in configuration 1). This establishes that the prioritized beam tracer provides a greater advantage in more complex geometries by tracing the more relevant beams first. On careful observation, we can notice that the gain of the prioritized beam tracer is not because of its enhanced performance in complex geometries, but rather due to the poorer performance of the other beam tracers. The wider spread of the breadth-first and EVERTims beam tracers in the complex apartment case than in the simplified apartment points to this fact. Nevertheless, the prioritized beam tracer significantly helps in dodging poor geometry.

In all the cases (Figs. 4.7 - 4.9), we see that the depth-first beam tracer performs most unpredictably, with the paths distributed over a wider region with no organization in terms of path energies. The histograms of the EVERTims beam tracer show organization of paths by energy similar to the breadth-first beam tracer, but only spread over a much wider region due to redundant tracing of beams that have been traced before. The breadth-first beam tracer shows a general organization in path energies, however, performs second best to the prioritized beam tracer which shows better compactness in the distribution of paths, organized mostly in the decreasing order of path energies regardless of the occlusion condition. Due to the unpredictability of results in the depth-first beam tracer, it cannot be reliably used in a real-time context. Since the EVERTims beam tracer is an adaptation of the depth-first beam tracer for a real-time context, we ignore the depth-first beam tracer and proceed with the EVERTims beam tracer in further analyses.

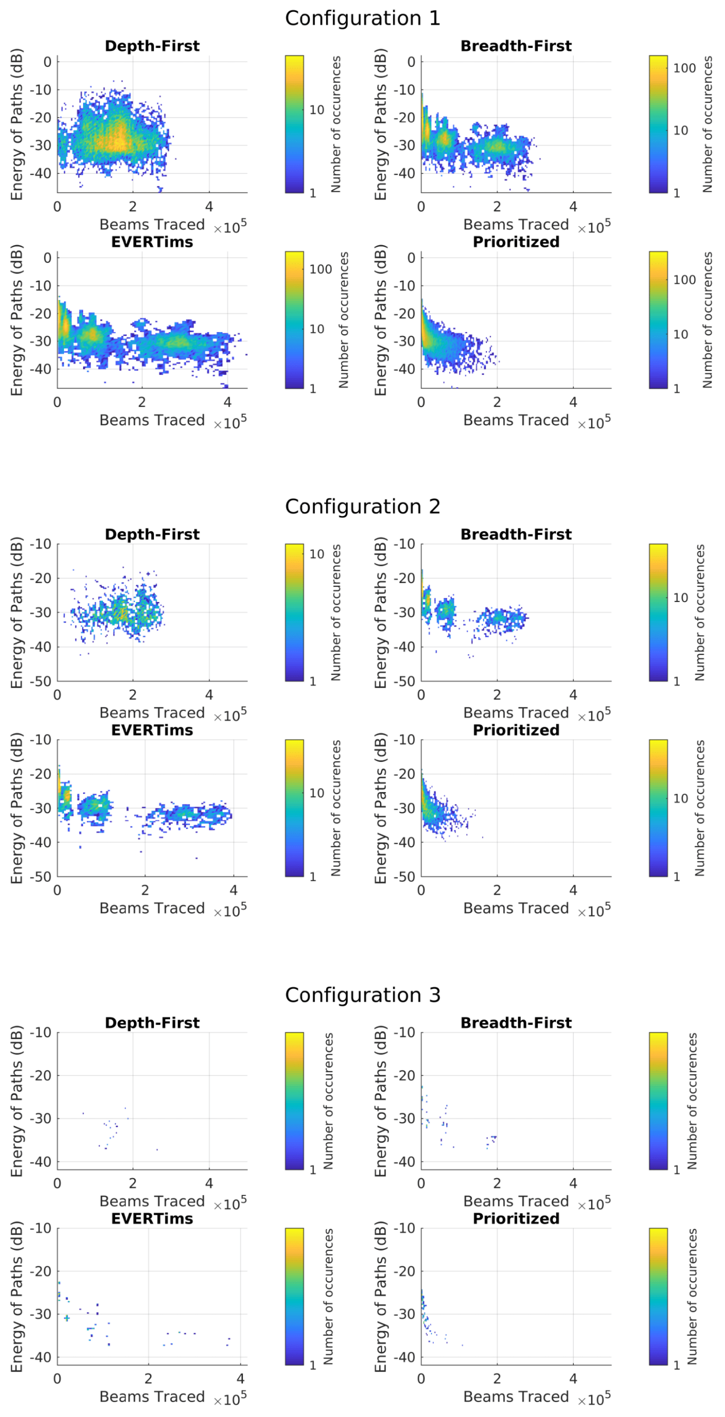


Fig. 4.8: Histograms of path energy vs. beam tree node index for the simplified apartment model.

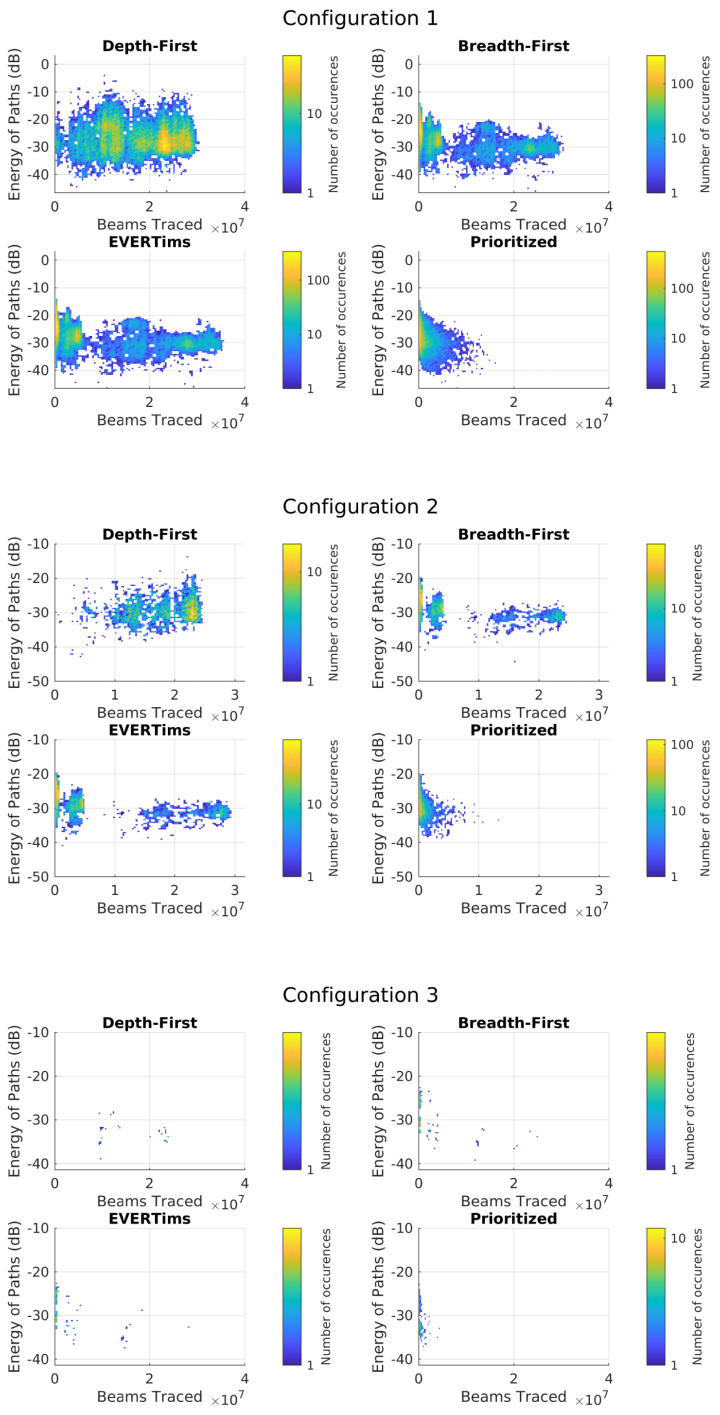


Fig. 4.9: Histograms of path energy vs. beam tree node index for the complex apartment model.

4.3.3 Simulation Cost

Due to the organization of paths in decreasing order of energy in the breadth-first, EVERTims and the prioritized beam tracers, we can expect the cumulative sum of the path energies to increase rapidly at first and then gradually approach the sum of all paths in the complete beam tree as more and more beams are traced (as predicted in Sec. 4.2). Figures 4.10 - 4.11 show the percentage cumulative energy of paths plotted against the number of beams traced for the three beam tracers in the different room models and occlusion configurations. The solid, coloured lines indicate the mean cumulative energy of paths for a given number of beams traced, for all trials in the said configurations, and the lightly shaded regions around them indicate the standard deviation in the cumulative path energies. The standard deviation is due to the inherent property of the beam tracers being oblivious to the listener position while tracing beams, making them listener-dynamic by requiring only a single beam tree for any listener position.

In Fig. 4.10 showing the percentage cumulative energy vs. number of beams traced in the cube shaped room model, we see that the prioritized beam tracer and the breadth-first beam tracer have very similar graphs, with the prioritized beam tracer beating the breadth-first beam tracer only marginally. The slight edge that the prioritized beam tracer has over the breadth-first beam tracer can be attributed to the better organization of paths according to their energies seen in Fig. 4.7. The curve of the EVERTims beam tracer appears to be similar to that of the breadth-first beam tracer, only stretched over a larger

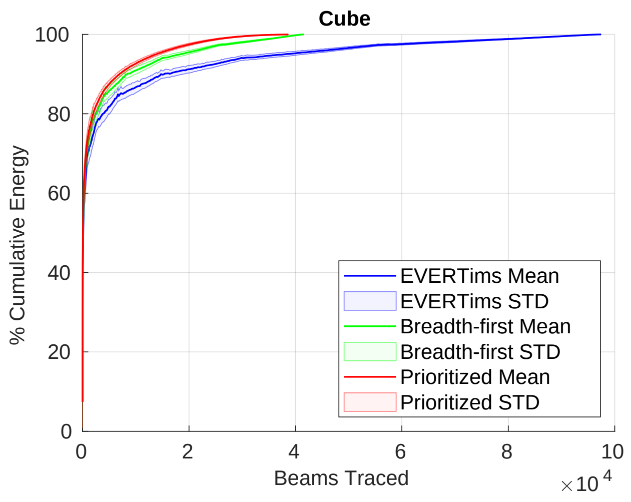


Fig. 4.10: Percentage cumulative path energy vs. beams traced - cube model.

number of beams as mentioned before.

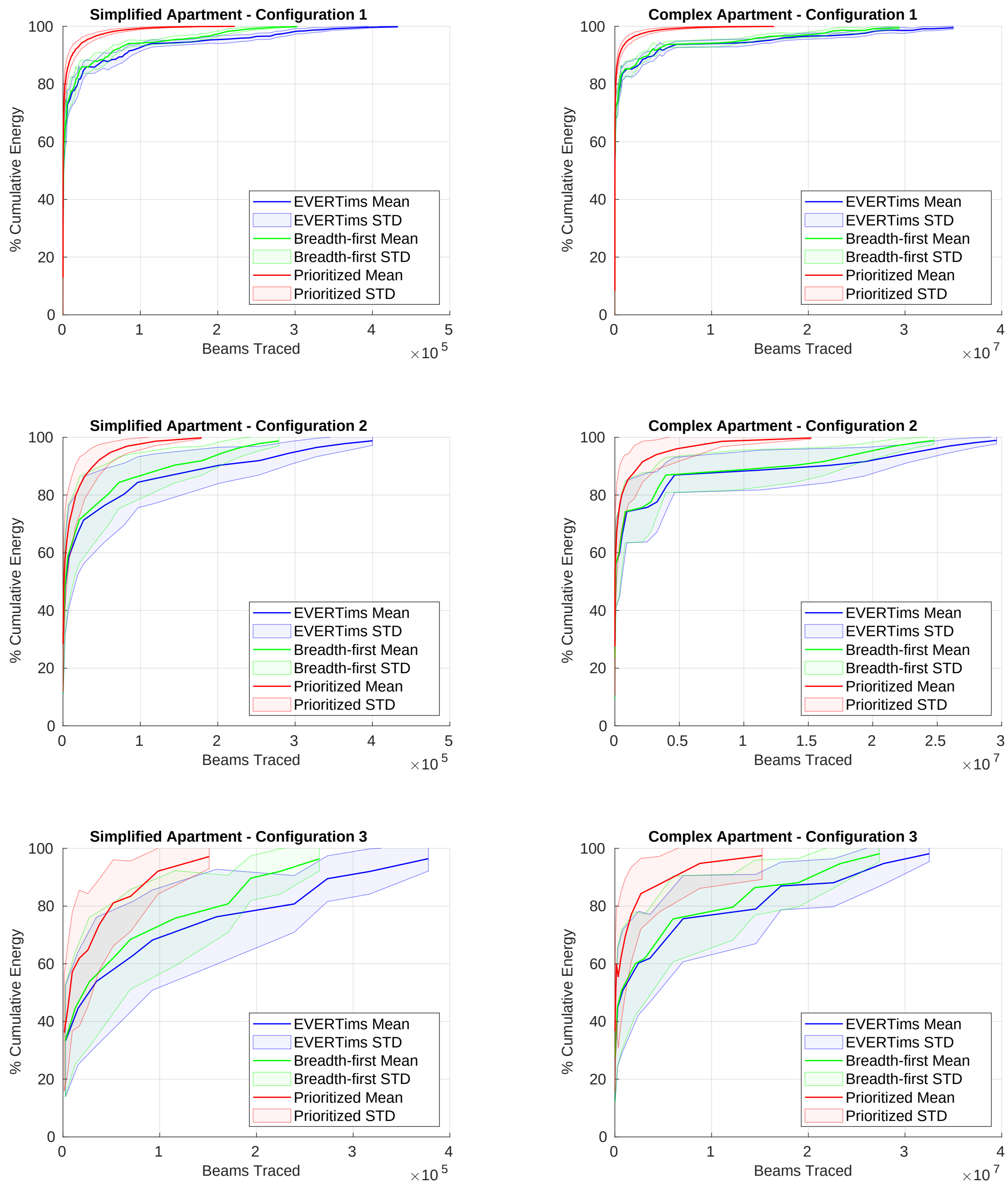


Fig. 4.11: Percentage cumulative path energy vs. beams traced - simplified apartment model (left) and complex apartment model (right).

In Fig. 4.11 showing the cumulative energy trend for the simplified and complex apartment models, we see that the prioritized beam tracer performs better than the other two beam tracers in all configurations of occlusion. The prioritized beam tracer reaches a plateau (saturates towards an upper bound) earlier than the other beam tracers and slowly approaches the maximum energy. As more and more occlusion is present between the source and the listener, the standard deviation increases. This can be seen by comparing the width of the shaded regions in each configuration of the apartment models. Configuration 1 has the least standard deviation, configuration 2 has a higher standard deviation and configuration 3 has the highest standard deviation among all configurations. Moreover, comparing the regions of standard deviation of the beam tracers within each case, we can see that the prioritized beam tracer has the least spread, which means its results are more deterministic than the other beam tracers. On comparing the curves obtained for the simplified apartment and the complex apartment models, we see that the graphs of the prioritized beam tracer reach the plateau significantly earlier than the other beam tracers in the complex apartment model, showing greater compactness. A similar observation was made from the histograms in Fig. 4.8 and Fig. 4.9, where the prioritized beam tracer showed better compactness in the complex apartment case than in the simplified apartment case.

Since the prioritized beam tracer recognizes the most significant paths before the less significant ones and the cumulative energy begins to saturate earlier than the other beam tracers, it would only be fair to compare the beam tracers at an intermediate stage accounting for the rate of increase, rather than comparing them based on the total number of beams needed to identify all paths. Hence, we use *Simulation Cost*, which we defined in Sec. 4.2 for comparison of the beam tracers, accounting in a fairly accurate way for the quickness in the increase of cumulative energy of paths. It is measured as the number of beams that must be traced in order to identify enough valid reflection paths with the energy sum being at least 90% of the total specular reflection energy. Table 4.1 shows a comparison of the average simulation costs for each algorithm in different room models and occlusion configurations. The costs given in the table are computed averaging the simulation cost of all trials in the given room and occlusion configuration.

The gain in simulation cost achieved by the beam tracers (given in Table 4.1) are obtained by comparing the mean simulation cost for a given room model and occlusion configuration with the mean simulation cost for the EVERTims beam tracer. The EVERTims beam tracer was chosen as the baseline for comparison since its cumulative path

Table 4.1: Average simulation costs of beam tracers in different room models.

		Cube	Simplified Apartment			Complex Apartment		
			Config. 1	Config. 2	Config. 3	Config. 1	Config. 2	Config. 3
EVERTims	Sim. Cost	24,779	113,675	258,502	367,331	5,421,903	18,052,826	26,919,969
	Gain	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Breadth-first	Sim. Cost	12,681	84,952	182,491	258,534	4,634,676	15,093,445	22,544,392
	Gain	1.95	1.34	1.42	1.42	1.17	1.20	1.19
Prioritized	Sim. Cost	7,111	11,362	38,761	74,330	489,811	1,982,258	4,950,708
	Gain	3.48	10.00	6.67	4.94	11.07	9.11	5.44

energy rises the slowest among the three beam tracers. We can see from the table that simulation cost gain of the prioritized beam tracer is the least for the cube model at 3.48 times, between 4.94 and 10 times for the simplified apartment model and between 5.44 and 11.07 times for the complex apartment model, whereas the gain in simulation cost achieved by the breadth-first beam tracer is only between 1-2 times, with the highest gain in the cube model and lowest gain in the complex apartment model. This proves that the prioritized beam tracer provides more gain in simulation cost with increasing complexity, although marginally, as observed before.

4.4 Termination of Beam Tracing

Rather than using a maximum reflection order, Figs. 4.7 - 4.11 show that we may be able to truncate the beam tree size (retaining the order in which the tree is populated) to reduce the amount of computations to perform depending on the level of accuracy required in each of the beam tracers. Among the three beam tracers, it is very clear that the prioritized beam tracer best gives us the ability to truncate the beam tree while maximizing the obtained total sum of path energies for any size of the beam tree. This shows that the prioritized beam tracer provides the most representative beam trees for a given number of beams traced. Therefore, a fixed number of beams may be used as a termination condition in the prioritized beam tracer. Even if termination of beam tracing process leads to loss of paths, the prioritized beam tracer guarantees that only the least significant paths are lost, hence maximizing the accuracy of the beam tree for a given number of beams traced. The curves for the prioritized beam tracer being above the curves of the other beam tracers in the Figs. 4.10 - 4.11 proves this fact.

However, in Table 4.1, we see that the simulation cost depends on the complexity of the room and the amount of occlusion. Even though the computation time for tracing a fixed number of beams across trials would be more or less the same for any room model, the accuracy of the obtained solutions are different for rooms of different complexities (Figs. 4.10 - 4.11 show this clearly). Hence, it might not be very straightforward to determine at what point to terminate the beam tracing process without performing offline tests.

A somewhat better approach to truncating the beam tree would be to trace all beams whose priorities are over a certain threshold. Applying a threshold on the priorities is equivalent to imposing a minimum requirement on the amount of energy that a beam must carry in order to be considered for beam tracing. Since beams that do not make the threshold are straightaway rejected, beam tracing would eventually come to a stop as there would be no more beams to trace. However, this is also essentially a truncation procedure, which would lead to a loss of accuracy in the overall output due to missing paths. Since the prioritized beam tracer produces the most representative beam trees for any given number of beams traced, it ensures that the captured specular reflection energy is maximized. Table 4.2 shows the sizes of the beam trees at different thresholds and the corresponding percentage cumulative path energies obtained, along with the time taken for simulation for different room models.

In Table 4.2, we can see that the increase in the number of beams traced is exponential in all cases until about -50 dB and slowly saturates beyond -75 dB. The large increase in the beginning is due to beam splitting caused by partial incidence of beams on surfaces, hence giving rise to more and more lower priority beams. Beyond a certain number of beams, when beams get narrower and narrower, the amount of beam splitting reduces and the number of beams begins to saturate. The difference in the cumulative energy obtained is also much larger in all room models and occlusion configurations in the first few thresholds than in lower thresholds. This again proves that the beam tracer traces the most significant beams earlier maximizing the number of paths being detected.

The cumulative energy reaches about 90% at a threshold of -37.5 dB and reaches near perfect results at -50 dB in most of the cases. Beyond this threshold, there is disproportionate increase in the cumulative energy for an increase in the number of beams traced. For example, in the complex apartment cases, in going from -50 dB threshold to -75 dB, there is an increase of 0.5 - 3.5% in cumulative energy for an increase in number of beams traced by 3 times (about 14 million more beams!). Such a disproportionate increase is seen even in the other cases, however not as exaggerated. Hence, choosing the right threshold could help in achieving optimum output at much lower costs.

Table 4.2: Average number of beams traced, percentage cumulative energy obtained and simulation times for different priority thresholds in the prioritized beam tracer.

Threshold	Average Measures	Cube	Simplified Apartment			Complex Apartment		
			Config. 1	Config. 2	Config. 3	Config. 1	Config.2	Config. 3
-12.5	**Number of Beams**	83.70	375.70	328.02	355.38	3,875.61	3,685.38	3,835.30
	% Cumulative Energy	51.84	63.00	37.11	17.46	54.73	40.99	27.43
	Solution Time (s)	0.000	0.001	0.001	0.001	0.013	0.013	0.013
-25	**Number of Beams**	1,003.04	3,550.33	3,264.08	3,547.32	42,741.61	36,312.30	45,249.77
	% Cumulative Energy	74.28	84.10	63.68	50.27	75.02	58.40	49.81
	Solution Time (s)	0.005	0.017	0.016	0.017	0.167	0.143	0.180
-37.5	**Number of Beams**	9,876.57	37,280.03	34,334.01	37,178.47	822,577.25	656,742.66	906,599.95
	% Cumulative Energy	92.61	97.03	90.61	84.12	93.41	83.78	79.60
	Solution Time (s)	0.054	0.220	0.205	0.215	3.814	3.057	4.264
-50	**Number of Beams**	32,735.83	146,825.88	140,751.66	145,162.12	7,017,823.47	6,122,469.36	7,285,519.25
	% Cumulative Energy	99.76	99.83	99.56	98.53	99.54	98.15	96.61
	Solution Time (s)	0.245	1.224	1.185	1.197	46.426	40.466	49.126
-75	**Number of Beams**	41,350.10	258,327.35	247,848.82	254,403.22	22,565,469.04	20,337,625.13	22,717,870.50
	% Cumulative Energy	100.00	99.97	99.99	100.00	100.00	100.00	99.99
	Solution Time (s)	0.400	3.253	3.106	3.156	304.304	277.603	313.534
-100	**Number of Beams**	41,485.25	270,695.32	259,549.21	266,219.83	24,142,881.29	21,791,106.58	24,259,454.10
	% Cumulative Energy	100.00	99.97	99.99	100.00	100.00	100.00	99.99
	Solution Time (s)	0.438	3.584	3.457	3.516	367.710	327.480	365.180
No Threshold	**Number of Beams**	41,553.20	290,778.85	278,204.44	285,592.13	25,931,870.72	23,448,231.99	26,017,842.30
	Solution Time (s)	0.495	4.449	4.258	4.350	414.557	377.540	423.954

In general, we can see that there is more predictability in the percentage cumulative energy obtained across room geometries (for similar occlusion conditions) for a given threshold than when using a fixed number of beams for termination. For example, comparing the case of the cube model with configuration 1 of the apartment models in Table 4.2, we see that the percentage cumulative energies of the paths that have been identified are similar. However, since the number of beams traced for a given threshold depends on the complexity of the room (number of polygons in the room model and amount of beam splitting that takes place), the computation time cannot be predicted, hence requiring offline tuning of the threshold for a desired computation time and accuracy of results.

Another important observation we can make is that, for any given threshold and occlusion configuration, the percentage cumulative energy obtained in the complex apartment case is almost always less than the simplified apartment model, although the dimensions of the models are similar. This is due to the fact that the complex model is made of a greater number of polygons which are smaller in size due to tessellations, hence causing the beams traced to be narrower with lower priorities, which are only traced at lower thresholds. Unnecessary tessellation of polygons in the room geometry could even lead to significant paths being missed at high priority thresholds. Hence, for optimum performance of the prioritized beam tracer, it is important to have larger polygons with as few tessellations as possible.

4.5 Timing Performance

4.5.1 Importance of Timing Analysis

So far, we have compared the performance of the beam tracers by the distribution of valid reflection paths in the obtained beam trees and the number of beams required to achieve 90% of the total specular reflection energy up to a specified reflection order. However, the figures and data do not comment on the timing performance of the beam tracers. In real-time applications of beam tracing, computational time of the algorithm plays a crucial role. Although it has been proven that the prioritized beam tracer captures valid reflection paths with fewer beams compared to the other beam tracers (which gives it a big advantage when used in an iterative refinement framework, like in the EVERTims package [13,14]), the data presented above does not prove that it performs any faster than the others. Since the beam tracers employ different methods to trace beams and different data structures to store them until they are traced, their computational costs are different and it is essential to compare them based on their timing performances.

The EVERTims beam tracer (the pseudo-breadth-first beam tracer) provides reflection paths in a breadth-first manner (one reflection order at a time) by constructing new beam trees for each order of reflection in a depth-first manner. The implementation uses recursive function calls to trace beams, which involves allocation of stack frames whose cost depends on the recursion depth. The breadth-first beam tracer constructs beam trees in a breadth-first manner using a simple queue to maintain a list of beams to be traced in a FIFO manner, whose computational cost increases linearly with number of beams traced. Finally, the prioritized beam tracer constructs beam trees in a best-first manner using a priority queue to maintain a list of beams in the order of their priorities, whose computational cost has a non-linear relationship with the number of beams already in the queue. This is due to the rearrangement of nodes in the queue required for the beam with the highest priority score to be available at the top on every insertion or deletion.

Figure 4.12 shows the time taken per beam by each beam tracer as the beam tree grows in a single trial in the complex apartment model with a maximum priority of beams at -100 dB. In the figure, we see that the time taken per beam stays constant in the breadth-first beam tracer, increases logarithmically for the EVERTims beam tracer and increases exponentially in the prioritized beam tracer. However, both the EVERTims and the prioritized beam tracers have regions where they are faster than the breadth-first beam tracer, but the EVERTims beam tracer is only faster for very low reflection orders. Hence, for the

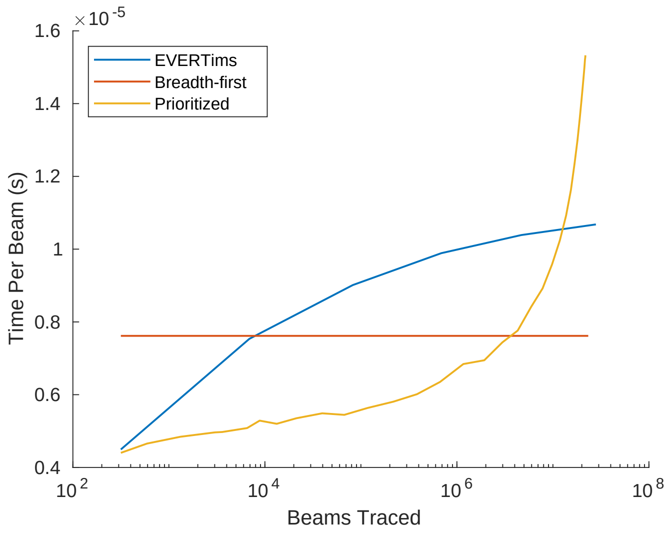


Fig. 4.12: Time taken per beam vs. Number of beams traced.

prioritized beam tracer to be more efficient than the breadth-first beam tracer, significant valid paths must be identified within the region where it is faster.

For real-time use, one may compute new solutions by performing beam tracing with hard-termination conditions as mentioned before (up to fixed number of beams or beams greater than a priority threshold, as mentioned in Sec. 4.3). However, we can see in Table 4.2 that under strict timing constraints (for real-time interactive application, an acceptable latency is 50 ms (Vorländer [45])), the beam tracer would result in tremendous loss of accuracy in complex room geometries. Another approach would be to exploit the source and room dynamics by continuing to perform beam tracing and fetching newly identified reflection paths in intervals, providing improvements in the results obtained until a completely new solution is required (when the source moves to a new location or the room geometry changes). Iterative order refinement employed in the EVERTims software package does exactly this, where the beam tracer provides bursts of reflection paths grouped by order to improve the accuracy of auralization. As explained in Sec. 2.4, the EVERTims beam tracer suffers from the drawback of redundant retracing of beams, which the breadth-first beam tracer does not suffer from since the leaf nodes are saved in a queue for continuation of beam tracing. Since the implementation of the prioritized beam tracer also supports saving of beams for future tracing, it may also be suitable for an iterative refinement framework. One may iteratively refine the solution by pausing the prioritized beam tracing process using an adaptive condition, such as an adaptive threshold which decreases in steps, or tracing a given number of beams before pausing to fetch new valid reflection paths.

In order to compare the timing performance of the beam tracers in a real-time iterative refinement framework, we perform beam tracing by iteratively refining the solution by order in the EVERTims and the breadth-first beam tracers and by priority in the prioritized beam tracer. For evaluation of the prioritized beam tracer, we use an adaptive threshold decreasing from 0 dB to -100 dB in steps of -2.5 dB to pause the beam tracing process and perform path validation. A hard-threshold of -100 dB is applied since the loss of accuracy is negligible in all room models and occlusion configurations (Table 4.2).

4.5.2 Simulation Time

Figures 4.13 - 4.15 show the percentage cumulative energy vs. computation time for all the room models and occlusion configurations. The staircase-like response of the beam tracers is because of pausing of the beam tracing process. The point where a step begins is the time when new reflection paths are available.

We can see in Fig. 4.13 showing the performance of the beam tracers in the cube room, that the breadth-first beam tracer and the prioritized beam tracer approach the maximum cumulative energy equally quickly and the EVERTims beam tracer is the slowest among the three. In the simplified and complex apartment models (Figs. 4.14 - 4.15), we can see

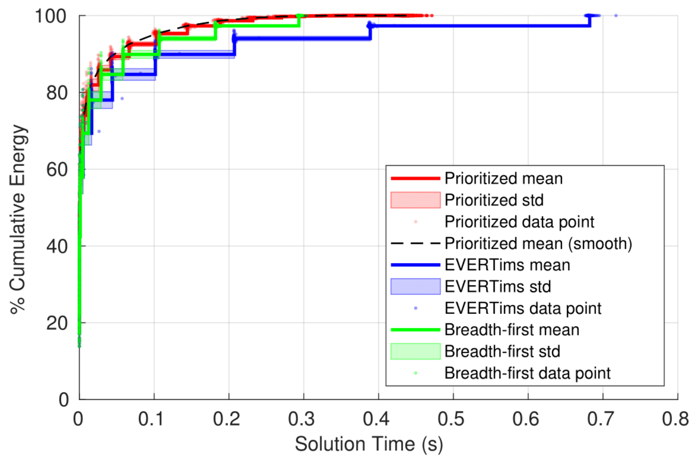


Fig. 4.13: Percentage cumulative path energy vs. time taken - cube model.

that the prioritized beam tracer performs better than the other two, reaching maximum energy the fastest. These curves imply that the prioritized beam tracer identifies significant reflection paths when its cost per beam is lower than that of the other beam tracers, in general. The lightly shaded regions around the mean curve (coloured solid lines) represent the standard deviation of the distribution at that time step. This is characteristic of listener-dynamic beam tracers, as seen in Sec. 4.3.

We can also observe that the refinement of the cumulative energy is the smoothest (i.e., the difference in cumulative energy between successive steps is small) for the prioritized beam tracer and is very coarse for the EVERTims and the breadth-first beam tracers since solutions are only available after tracing all beams up to a given reflection order. The refinement in the prioritized beam tracer may be made smoother by reducing this adaptive threshold step size or even tracing a small number of beams before pausing (adaptive number of beams). Figures 4.13 - 4.15 show the smoothest refinement in the percentage cumulative energy in dashed black lines.

However, we can observe that in simpler geometries the prioritized beam tracer takes a longer time than the breadth-first beam tracer to reach the complete solution, and in the complex apartment cases, we see that the prioritized beam tracer takes more time than both the breadth-first and the EVERTims beam tracers. This is because of the paths that are identified in beams of very low priorities and the exponential increase in the cost per beam as the beam tree grows, hence being detected very late. However, since these paths are of very low energy, they often do not contribute significantly to the total specular reflection energy. Hence, the most important observation here is that, in all the cases, the prioritized beam tracer prevails over the other beam tracers in approaching the maximum energy the fastest.

As with the percentage cumulative energy vs. number of beams traced (Figs. 4.10 - 4.11), the standard deviation in computation times are proportional to the amount of occlusion. We can see in the graphs for the simple and complex apartment cases (Figs. 4.14 - 4.15) that, for more occlusion between the source and the listener, there is greater standard deviation in the percentage cumulative energy of the reflection paths vs. time taken. More occlusion means more failed path validations in the earlier parts of the beam tree. Since the beam tree gets deeper as beam tracing continues, there is more and more unpredictability in path validations since the low priority beams have lower likelihood of detecting reflection paths. Hence, we can say that the prioritized beam tracer works best when there is little to no occlusion, however works better than the other beam tracers even

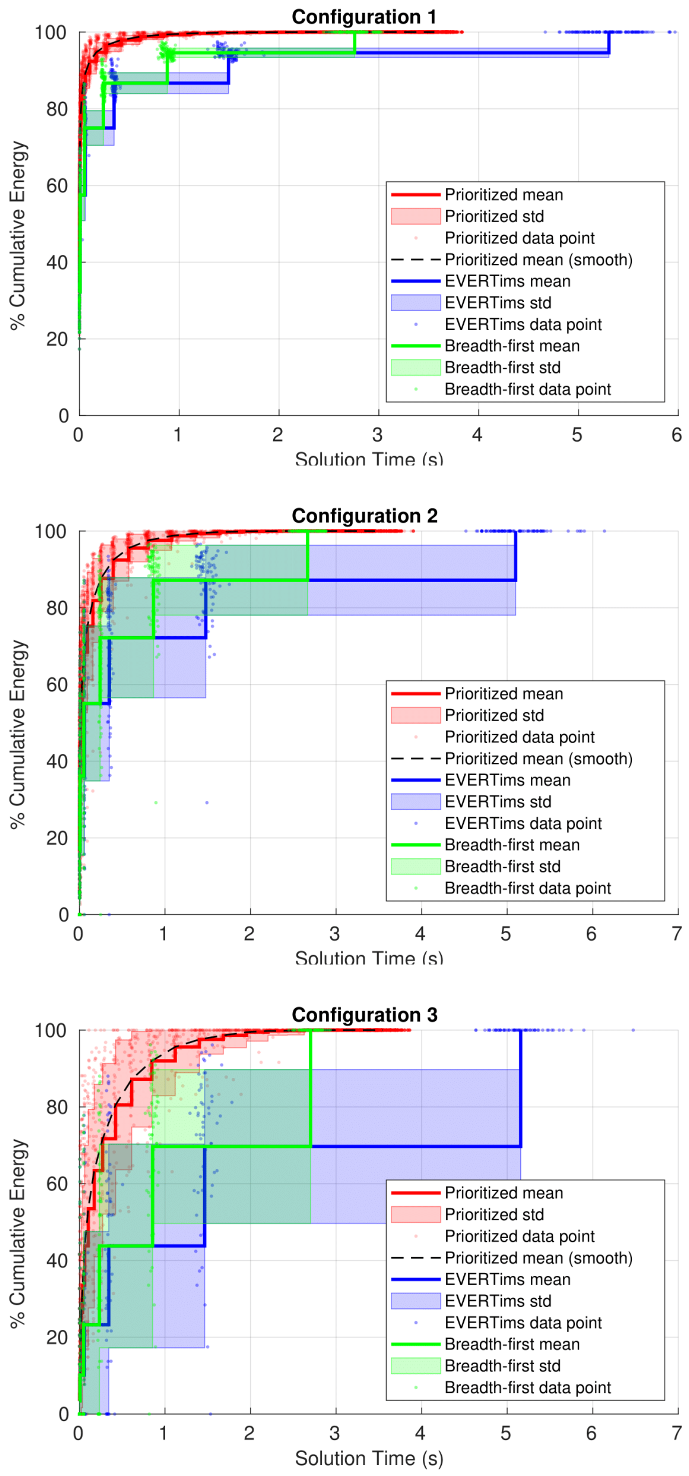


Fig. 4.14: Percentage cumulative path energy vs. time taken - simplified apartment model.

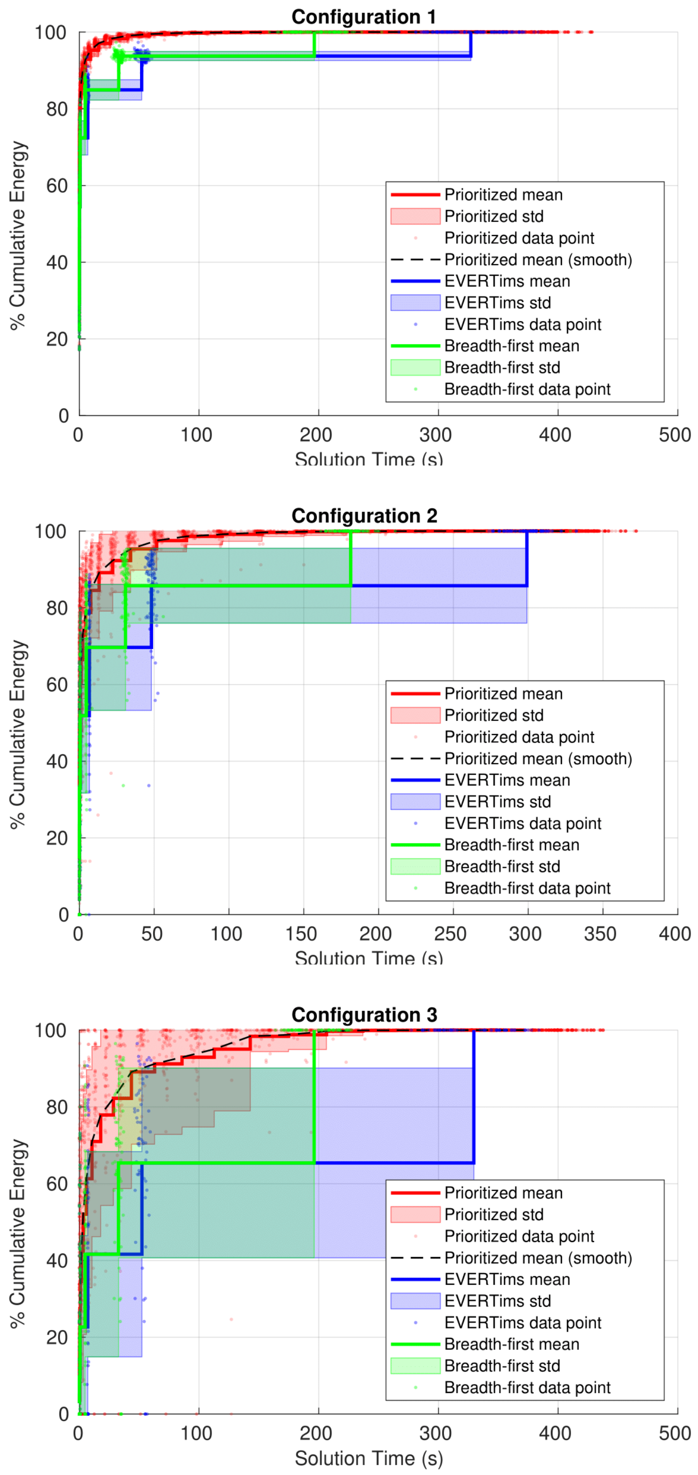


Fig. 4.15: Percentage cumulative path energy vs. time taken - complex apartment model.

in highly occluded conditions. Even within each occlusion configuration, the standard deviations of the EVERTims and the breadth-first beam tracer are far higher compared to the prioritized beam tracer.

Now, to compare the beam tracers based on computation time, we use a metric similar to the *Simulation Cost* used in Sec. 4.3, called the *Simulation Time*. We defined *Simulation Time* in Sec. 4.2 as the time taken by the beam tracer to detect paths whose energies add up to at least 90% of the total specular reflection energy. Since the priority queue allows us to pause the beam tracing process at any time while assuring that the maximum possible number of significant paths are detected, we measure simulation times for the prioritized beam tracer as the time taken to trace enough number of beams to push the cumulative energy over the 90% mark, by interpolating between the two nearest points for which timing information is available, for each trial. Since the same is not possible in the breadth-first and the EVERTims beam tracers, we compute the simulation time as the time taken to compute enough orders of reflection for the sum of path energies to reach at least 90% of the total. Table 4.3 shows the simulation times for each of the simulated room models and occlusion configurations.

Table 4.3: Simulation times of the beam tracers for different room models

		Cube	Simplified Apartment			Complex Apartment		
			Config. 1	**Config. 2**	**Config. 3**	**Config. 1**	**Config. 2**	**Config.3**
Evertims	**Sim. Time (s)**	0.17	1.45	3.30	4.65	57.14	191.97	283.80
	Gain	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Breadth-first	**Sim. Time (s)**	0.09	0.81	1.75	2.44	35.07	117.59	169.03
	Gain	1.94	1.80	1.88	1.90	1.63	1.63	1.68
Prioritized	**Sim. Time (s)**	0.05	0.07	0.30	0.65	2.99	12.87	43.72
	Gain	3.75	19.97	11.11	7.17	19.14	14.91	6.49

The gain in simulation times for each of the beam tracers given in Table 4.3 is computed in comparison with the simulation times of the EVERTims beam tracer. We see that the EVERTims beam tracer performs the poorest, taking the longest time to compute 90% of the total specular reflection energy up to the given maximum reflection order, the breadth-first beam tracer shows a gain in simulation time around 2 times, and the prioritized beam tracer performs the best with computing the solution 3.75 to 20 times faster than the EVERTims beam tracer on average for different room complexities and occlusion configurations.

We can see from Table 4.3 that the gain in simulation times are comparable for the two apartment models and in some cases in the complex apartment model, the prioritized beam tracer under-performs compared to the simplified apartment model. This is also apparent in the graphs of percentage cumulative path energies vs. time (Figs. 4.13 - 4.15), where there is not much that differentiates between the simplified and the complex apartment models. This is different from what we saw in Table 4.1 where the gain in simulation cost was slightly higher in the complex apartment model. This could be due to the fact that there are roughly 6 times more polygons in the complex apartment model compared to the simplified model, with the beam trees being much wider. Also, as we saw in Fig. 4.12, the computational time per beam increases exponentially as the beam tree grows. Hence, for wider beam trees in more complex room geometries, the computational performance of the prioritized beam tracer drops. This again points to the fact that simpler geometries with less tessellations are more favourable. However, further tests may be required to evaluate the gain in performance for even higher complexities in room geometry. The scope of this thesis ends at the complexity of the complex apartment model and further complexities are left for future work.

In hindsight, since the complex apartment model contains unnecessary tessellations, causing the reflecting polygons to be smaller, the priorities of the beams traced would naturally be lower. Apart from being smaller in value, the priorities of the beams competing would also be similar due to the high density of beam splitting. When beams of similar priorities compete, there could potentially be a drop in performance of the priority queue due to increased number of rearrangements on every insertion and deletion. This could also be the reason for the exponential increase in the computation time per beam as the beam tree grows, which was seen in Fig. 4.12. An implementational hack to improve the performance of insertion into and deletion from the priority queue in the context of beam tracing would be to have a thread running in parallel that takes care of the priority queue management without blocking the geometrical computations of beam tracing. The exploration of such implementations is also left for future work.

4.5.3 Use of Hard-Threshold

Table 4.3 shows the performance of the prioritized beam tracer when a hard-threshold of -100 dB is applied. However, depending on the accuracy of output and beam tree size requirements, one may choose a lower hard-threshold to reduce the overall computation time (see Table 4.2), along with an adaptive pausing condition. The advantage of doing so is that there would be less number of beams saved in the priority queue, hence lower cost

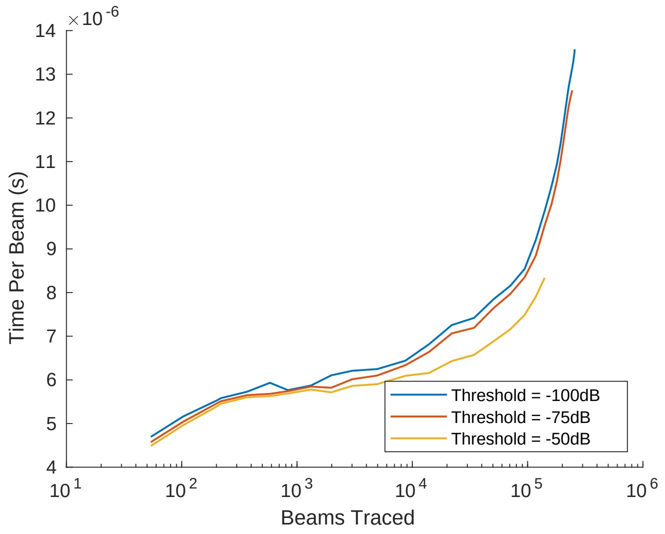


Fig. 4.16: Time taken per beam vs. Number of beams traced for different thresholds.

of priority queue maintenance. Figure 4.16 shows the time taken per beam for different hard-thresholds for a single trial in the complex apartment model. We can see that, for the same number of beams traced, the computation time per beam is smaller for higher values of hard-thresholds. This is due to the fact that fewer beams are saved in the priority queue for future beam tracing, thereby reducing the priority queue maintenance cost. However, one should bear in mind the trade-off between computation time and accuracy due to a greater loss of accuracy at higher thresholds (due to smaller beam trees) while choosing the hard-threshold value.

4.6 Tunability of the Prioritized Beam Tracer

The histograms in Figs. 4.7 - 4.9 and the percentage cumulative energy plotted against number of beams traced in Figs. 4.10 - 4.11 show that the beam tree can be truncated without significant loss in the accuracy of results. Figures 4.13 - 4.15 show that terminating the beam tracing process at any time would result in best possible accuracy of results when compared to the other beam tracers. This property of the beam tracer yielding maximum accuracy of results for any given beam tree size and at any given point in time comes from the fact that the prioritized beam tracer maximizes effort in directions where it makes most sense using the knowledge of the beam priority scores, hence identifying significant specular reflection paths earlier.

Therefore, the prioritized beam tracer may be tailored to suit a wide variety of needs due to its tremendous fine tuning capabilities. In an offline process where timing is not of concern, one may choose a very low hard priority threshold for a higher degree of accuracy. For a real-time application where timing is key, one may choose a high enough hard priority threshold and a fine iteration interval (by varying the adaptive threshold step size or choosing a smaller number of beams to trace) in order to minimize latency. In either case, one can confidently rely on the beam tracer to provide the best possible results under the given constraints of accuracy and latency.

4.7 Influence of the Prioritization Factors on the Performance

From our initial definitions of the priority factors, we saw that the beam-width factor determines the likelihood of finding valid paths within a beam and the other factors concern the perceptual relevance of the beams, hence determine the likelihood of finding perceptually significant reflection paths in a beam. We see from the interpretation of the the priority factors in Sec. 3.3 that the absorption-based priority factors only make sense when used with the beam-width factor, since it is important to know the starting energy in order to determine the energy of a beam after attenuation. However, in order to understand the extent to which the attenuation related priority factors (the air-absorption factor and the material-absorption factor) actually influence the overall results of the prioritized beam tracer, we perform brief tests comparing four different versions of the prioritized beam tracer that use different combinations of the priority factors to perform beam tracing:

1. **Prioritization with beam-width factor only:** In this version, beams are prioritized based on the energy emitted by the source within the region of the given beam. Only the possibility of a valid reflection path is considered and their perceptual significance is ignored.

2. **Prioritization with beam-width factor and air-absorption factor:** In this version, beams are prioritized based on the energy incident on the reflecting polygons after air-absorption, giving more importance to beams that have incurred smaller air-absorption losses.

3. **Prioritization with beam-width factor and material-absorption factor:** In this version, beams are prioritized based on the energy reflected by surfaces without considering air-absorption.

4. **Prioritization with beam-width factor, air-absorption factor and material-absorption factor:** In this version, beams are prioritized based on the energy reflected by surfaces while taking air-absorption into account.

Considering the room dimensions and the geometries, our prediction would be that, among the two absorption priority factors, the more important factor influencing the performance of the prioritized beam tracer would be the material-absorption factor, since the chosen materials are fairly absorptive and the dimensions of the room geometry are not quite big enough for air-absorption to make a significant difference. Table 4.4 shows a comparison of the simulation cost and simulation time gains of the different versions of the prioritized beam tracer in different room geometries and occlusion configurations in comparison with a prioritization with only the beam-width factor.

Table 4.4: Comparison of performance of prioritization methods. The simulation cost and simulation time gains achieved by the prioritization techniques are computed with respect to the results of the prioritized beam tracer with beam-width factor only. BW - beam-width factor, Air - air-absorption factor, Mat - material-absorption factor.

Prioritization	**Gain**	**Cube**	**Simplififed Apartment**			**Copmlex Apartment**		
			Config. 1	**Config. 2**	**Config. 3**	**Config. 1**	**Config. 2**	**Config. 3**
BW	**Sim. Cost**	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Sim. Time	1.00	1.00	1.00	1.00	1.00	1.00	1.00
BW, Air	**Sim. Cost**	1.04	1.05	1.05	1.01	1.07	1.06	1.01
	Sim. Time	1.05	1.09	1.09	1.05	1.11	1.10	0.96
BW, Mat	**Sim. Cost**	1.05	1.10	1.10	1.06	1.38	1.23	1.05
	Sim. Time	1.07	1.10	1.10	1.26	1.40	1.40	1.09
BW, Air, Mat	**Sim. Cost**	1.08	1.13	1.13	1.07	1.45	1.29	1.07
	Sim. Time	1.30	1.32	1.32	1.35	1.88	1.72	1.16

The table shows that in all cases, the material-absorption factor has a higher influence on the results of the prioritized beam tracer than the air-absorption factor, with the material-absorption factor improving the simulation cost by 5 - 38% and simulation time by 7 - 40% over using only the beam-width factor, and the air-absorption factor only improving the simulation cost by 1 - 7% and simulation time by 5 - 11%. Together, however, they improve the simulation time by 16 - 88% depending on the room geometry and level of occlusion, the reason for which is not very evident from the available data. Moreover, the room geometries considered in this thesis are not sufficient to come to a conclusion on the amount of influence of the priority factors, which requires evaluation with a variety of rooms with different dimensions and material-absorption properties. A thorough investigation in this respect is left for future work. Such a study would shed light on the application and tunability of the thresholds for individual priority factors depending on the nature of the room, in order to further improve the performance of the prioritized beam tracer.

Although a discernible pattern is not clear from the results presented, it would be safe to conclude that the order of importance of the priority factors is:

1. Beam-width factor
2. Material-absorption factor
3. Air-absorption factor

The reason for the beam-width factor to be the most important factor is trivial since it determines the basic likelihood of existence of a valid reflection path within a beam. The other two factors may depend on the room geometry and absorption characteristics of the reflection surface, hence cannot be commented on. However, in the room models used for simulation in this thesis, it is clear that the material-absorption factor is of more consequence than the air-absorption factor. Since the inclusion of the two absorption based factors does not significantly alter the computational costs (refer Sec. 3.3), we might as well use them to improve the effectiveness of the prioritized beam tracer.

In the next chapter, we provide concluding remarks and discuss possible future directions of the project.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, a physically-informed prioritized beam tracer was presented that judiciously traces beams based on the energies that they carry to efficiently detect specular reflection paths from the source to the listener in a given virtual environment. The main objective was to detect the more significant reflection paths earlier in order to reduce the number of beams to trace, while preserving its tolerance for listener movement, thereby reducing the computational cost of the beam tracing algorithm. The presented beam tracer uses three factors for prioritization of beams: the beam-width factor, the air-absorption factor and the material-absorption factor. Together, the three priority factors signify the energy of beams and indicate the likelihood of detecting perceptually significant specular reflection paths.

The performance of the prioritized beam tracer was evaluated in three different room models of different complexities containing different number of reflecting surfaces: a cube (a simple room model with 6 polygons), a simplified apartment (a mid-complexity model with 53 polygons) and a complex apartment (a complex model with 315 polygons) in three different occlusion configurations: no occlusion, light occlusion and heavy occlusion. The prioritized beam tracer was compared with a depth-first beam tracer, a breadth-first beam tracer and pseudo-breadth-first beam tracer (used in the EVERTims real-time room acoustic simulator) based on distribution of paths in the beam tree and time taken for detecting significant amount of specular reflection energy. Different termination conditions for the prioritized beam tracing, such as a fixed number of beams, a fixed priority threshold and a hybrid method with adaptive pausing conditions along with a fixed hard-threshold, were also explored in this thesis.

The prioritized beam tracer shows compactness in distribution of paths in the beam tree in all simulation cases, i.e., the paths are distributed in a small region in the beginning of the beam tree, with the paths being organized mostly in the order of decreasing path energies, regardless of the reflection order. Hence, the most significant paths were detected with fewer beams compared to the other beam tracers. On analysis of the simulation results, it was found that the prioritized beam tracer traces 90% of the specular reflection energy with 3.48 times less beams compared to the EVERTims beam tracer and 1.79 times less beam compared to the breadth-first beam tracer in the simple room model, 4.94 to 10 times less beams compared to the EVERTims beam tracer and 3.48 to 7.46 times compared to the breadth-first beam tracer in the mid-complexity room model depending on the level of occlusion, and 5.44 to 11.07 times less beams compared to the EVERTims beam tracer and 4.57 to 9.46 times compared to the breadth-first beam tracer in the complex room model. This proves that the prioritized beam tracer most efficiently traces beams in any room model and occlusion configuration while preserving the tolerance for listener movements, performing better in more complex rooms with poor geometries.

The prioritized beam tracer was also evaluated for its suitability in a real-time context with iterative refinement of the obtained specular reflection energy with time. Since the implementation of the prioritized beam tracer allows for pausing the beam tracing process and resuming later, it suits the iterative refinement framework the best among all the considered beam tracers. In spite of added costs of priority computation and management of priority queue, the prioritized beam tracer was found to perform the best in terms of computation time, owing to its efficient selection of beams for tracing. The prioritized beam tracer performs 3.75 times faster than the EVERTims beam tracer and 1.93 times faster than the breadth-first beam tracer in the simple room model, 7.17 to 19.97 times faster than the EVERTims beam tracer and 3.77 to 11.1 times faster than the breadth-first beam tracer in the mid-complexity model depending on the level of occlusion, and 6.49 to 19.14 times faster than the EVERTims beam tracer and 3.86 to 11.74 times faster than the breadth-first beam tracer in the complex room model. Even though the prioritized beam tracer performs faster than the other considered beam tracers, all results point to the fact that simpler and well constructed rooms with fewer tessellations and bigger reflecting polygons suit real-time simulation better.

Since the prioritized beam tracer identifies the more significant paths before the less significant ones (generally), it ensures that the accuracy of the specular reflection energy detected is maximized for any given number of beams traced as opposed to the other beam tracers, which perform poorer in this respect. Results presented in this thesis show that the physically informed prioritized beam tracer can be fine tuned based on the computational requirements and the acceptable level of accuracy, maximizing the performance under the given constraints for any given application on any computing platform. The reduced computational cost of beam tracing achieved by the presented beam tracer could increase the tolerance on the rate of source movement while maximizing the accuracy of obtained results.

5.2 Future Work

The prioritized beam tracer has been evaluated in three different geometries with the most complex geometry being a complex apartment, with tessellations and thick walls. From the results, we can see that the performance gain seems to saturate at some point. It would be interesting to evaluate the performance of the proposed beam tracer in more complex room geometries such as auditoria and concert halls where there is minimum occlusion, and buildings with several chambers and perhaps even cityscapes where there is heavy occlusion. This would also give us an opportunity to investigate the influence of the different prioritization factors in environments of different sizes and material-absorption properties, hence could help in fine tuning the performance of the beam tracer according to the environment.

The source code of the beam tracing library used in this thesis contains many segments that could be optimized further in order to improve the timing performance of the beam tracer. Several parts of the code can be parallelized to improve its computation speed. For example, in Chapter 4, we saw that the cost of priority queue maintenance increases exponentially as the beam tree grows. In the current implementation, priority queue operations such as insertion and deletion (with internal rearrangement of nodes in the queue) block the functioning of the rest of algorithm which may not be dependent on the priority queue, like priority computations of the other child beams, beam casting and frustum culling for tracing the next beam, etc. Similarly, path validation is performed only after pausing the beam tracing process. However, path validation may be performed simultaneously as beams are being traced in a multi-threaded implementation where the beam tracer and the path validator run on different threads, hence achieving very fine refinement of the solution. Path validation for beam tree node buckets (skip-sphere optimization) may also be done in parallel since no two buckets have common data between them.

Apart from these implementational improvements, some amount of speed up may also be achieved by using faster techniques and data structures to store untraced beams. One way would be to use a weak prioritization technique to reduce the cost of rearrangement of beams in the priority queue on every insertion and deletion, which arises due to the strict imposition of order of beam tracing. This may be done by storing beams in a finite number of bins corresponding to different predetermined ranges of beam priorities. These bins may then be stored in an ordered linked list or a priority queue and the beam to be traced at every iteration of the beam tracing step may be chosen at random or by order of insertion from highest priority bin. However, the choice of number of bins may depend on several factors such as complexity of room geometry, size of the room, required performance efficiency, etc., which may not be trivial to determine.

Naturally occurring sound sources have complex 3D radiation characteristics. In Chapter 3, directional radiation properties of a source was presented as a potential factor for prioritization of beams. During the course of this project, some techniques of harnessing the radiative properties were explored and work in this respect is currently in progress. However, it was not reported since it is out of the scope of this thesis. The ideal way to go about including the source directivity would be to compute the exact energy incident on a reflecting surface depending on the orientation of the source. This involves computation of the surface integral of the source directivity function within the boundary of projection of the surface on to a unit-sphere centered at the source. It was found to be highly inefficient as the computational costs blew out of proportion within a few thousand beams, mainly due to high-order spherical harmonics functions. This solution would also increase the computational cost for a rotating source since the beam priorities must be recomputed each time the source rotates.

The solution that is currently being pursued involves using imaginary platonic solids (regular polyhedrons with equal edge lengths and convex faces) around the source for applying directional weights. This technique provides a crude way of prioritizing beams in directions where there is maximum energy emitted by the source while retaining its rotational dynamics. In the technique, we assign weights to each face of the platonic solid based on the surface integral of the 3D source directivity function over the azimuth and elevation limits of the face, which is applied to the priorities of beams being emitted through it. The surface integral is computed as the dot prod of the 3D radiation function and the face of the platonic solid in the spherical harmonic domain. Computation of directive weights is done on a separate thread in order to avoid computational loading by the intensive mathematical operations in the spherical harmonic domain. Complete details of the algorithm and implementation will be presented in a future publication.

The beam tracing algorithm constructs beam trees that are used to identify only specular reflection paths and sometimes low-order diffraction paths as mentioned in [9, 23]. There have rarely been any attempts at modeling diffuse reflections within the framework of beam tracing and room acoustic simulators usually rely on other methods, such as ray tracing or other stochastic techniques like in [14]. However, the beam trees contain a lot more information such as the mutual visibility of reflecting polygons which could be exploited to model other acoustic phenomena such as diffuse reflections and scattering.

Finally, we saw in Sec. 4.4 that more complex geometries and poorly constructed rooms weigh down the performance of the beam tracer, rendering it useless in real-time applications. This is true even for other techniques such as acoustic radiance transfer (ART). Hence, more research in the direction of automatic adjustment of level of detail of the room such as the ones used in computer graphics rendering may help in making acoustic simulation algorithms more efficient for real-time purposes. Since human audition is far less sensitive than human vision, acoustic simulations with very low resolution but adequate detail, may be sufficient for believably realistic rendering of virtual acoustics.

References

[1] S. Laine, S. Siltanen, T. Lokki, and L. Savioja, "Accelerated beam tracing algorithm," *Applied Acoustics*, vol. 70, no. 1, pp. 172–181, 2009.

[2] M. R. Schroeder and B. F. Logan, ""Colorless" artificial reverberation," *The Journal of the Acoustical Society of America*, vol. 32, no. 11, pp. 1520–1520, 1960.

[3] J. Moorer, "About this reverberation business," *Computer Music Journal*, vol. 3, pp. 605–639, 01 1985.

[4] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Audio Engineering Society Convention 90*, Feb 1991.

[5] A. Reilly and D. McGrath, "Convolution processing for realistic reverberation," in *Audio Engineering Society Convention 98*, Feb 1995.

[6] S. Bilbao, K. Arcas, and A. Chaigne, "A physical model for plate reverberation," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, pp. V – V, 06 2006.

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

[8] N. Raghuvanshi, J. Snyder, R. Mehra, M. Lin, and N. Govindaraju, "Precomputed wave simulation for real-time sound propagation of dynamic sources in complex scenes," in *ACM Transactions on Graphics*, vol. 29, p. 68, ACM, 2010.

[9] T. Funkhouser, P. Min, and I. Carlbom, "Real-time acoustic modeling for distributed virtual environments," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Technology*, SIGGRAPH '99, (New York, NY, USA), pp. 365–374, ACM Press/Addison-Wesley Publishing Co., 1999.

[10] F. Antonacci, M. Foco, A. Sarti, and S. Tubaro, "Fast tracing of acoustic beams and paths through visibility lookup," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 812–824, May 2008.

[11] F. Antonacci, A. Sarti, and S. Tubaro, "Two-dimensional beam tracing from visibility diagrams for real-time acoustic rendering," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, p. 642316, Sep 2010.

[12] D. Marković, F. Antonacci, A. Sarti, and S. Tubaro, "3D beam tracing based on visibility lookup for interactive acoustic modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, pp. 2262–2274, Oct. 2016.

[13] M. Noisternig, B. F. G. Katz, S. Siltanen, and L. Savioja, "Framework for Real-Time Auralization in Architectural Acoustics," *Acta Acustica united with Acustica*, vol. 94, no. 6, pp. 1000–1015, 2008.

[14] D. Poirier-Quinot, M. Noisternig, and B. F. G. Katz, "Evertims: Open source framework for real-time auralization in VR," in *Proc. of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences, London, United Kingdom, Aug. 23 - 26, 2017*, pp. 34:1–34:5, 2017.

[15] L. C. Sutherland, J. E. Piercy, H. E. Bass, and L. B. Evans, "Method for calculating the absorption of sound by the atmosphere," *The Journal of the Acoustical Society of America*, vol. 56, no. S1, pp. S1–S1, 1974.

[16] C. Hendrix and W. Barfield, "The sense of presence within auditory virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 5, pp. 290–301, Jan. 1996.

[17] M. Kleiner, B.-I. Dalenbäck, and P. Svensson, "Auralization-an overview," *Journal of the Audio Engineering Society*, vol. 41, no. 11, pp. 861–875, 1993.

[18] W. Zhang, P. N. Samarasinghe, H. Chen, and T. D. Abhayapala, "Surround by sound: A review of spatial audio recording and reproduction," *Applied Sciences*, vol. 7, no. 5, 2017.

[19] S. Siltanen, T. Lokki, and L. Savioja, "Rays or waves? Understanding the strengths and weaknesses of computational room acoustics modeling techniques," in *the Internation Symposium on Room Acoustics (ISRA2010), Melbourne, Australia, Aug. 29-31, 2010.*

[20] M. Vorländer, *Simulation models*, pp. 147–173. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[21] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

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

[23] V. Pulkki, T. Lokki, and L. Savioja, "Implementation and visualization of edge diffraction with image-source method," in *Audio Engineering Society Convention 112*, Apr 2002.

[24] A. Krokstad, S. Strom, and S. Sørsdal, "Calculating the acoustical room response by the use of a ray tracing technique," *Journal of Sound and Vibration*, vol. 8, no. 1, pp. 118 – 125, 1968.

[25] H. Kuttruff, "Simulated reverberation curves in rectangular rooms with diffuse sound fields," *Acta Acustica united with Acustica*, vol. 25, pp. 333–342, 12 1971.

[26] H. Kuttruff and T. Straßen, "Zur abhängigkeit des raumnachhalls von der wanddiffusität und von der raumform" ("on the dependence of reverberation time on the 'wall diffusion' and on room shape")," *Acustica*, vol. 45, p. 246, 1980.

[27] D. Schröder and A. Pohl, "Modeling (non-)uniform scattering distributions in geometrical acoustics," *Proceedings of Meetings on Acoustics*, vol. 19, no. 1, p. 015112, 2013.

[28] M. Mehta and K. Mulholland, "Effect of non-uniform distribution of absorption on reverberation time," *Journal of Sound and Vibration*, vol. 46, no. 2, pp. 209 – 224, 1976.

[29] C. Christensen and J. Rindel, "A new scattering method that combines roughness and diffraction effects," *The Journal of the Acoustical Society of America*, vol. 117, no. 4, pp. 2499–2499, 2005.

[30] H. Lehnert, "Systematic errors of the ray-tracing algorithm," *Applied Acoustics*, vol. 38, no. 2, pp. 207 – 221, 1993.

[31] M. Vorländer, "Die genauigkeit von berechnungen mit dem raumakustischen schallteilchenmodell und ihre abhängigkeit von der rechenzeit" ("the accuracy of calculations using the room acoustical ray-tracing-model and its dependence on the calculation time")," *Acustica*, vol. 66, p. 90, 1988.

[32] M. Vorländer, "Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm," *The Journal of the Acoustical Society of America*, vol. 86, no. 1, pp. 172–178, 1989.

[33] D. van Maercke and J. Martin, "The prediction of echograms and impulse responses within the epidaure software," *Applied Acoustics*, vol. 38, no. 2, pp. 93 – 114, 1993.

[34] 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, (New York, NY, USA), pp. 21–32, ACM, 1998.

[35] 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, (New York, NY, USA), pp. 545–552, ACM, 2001.

[36] "Virtual choreographer (virchor). software released under gpl 2008." `http://virchor.wiki.sourceforge.net/`. Deprecated.

[37] D. G. Malham and A. Myatt, "3-d sound spatialization using ambisonic techniques," *Computer Music Journal*, vol. 19, no. 4, pp. 58–70, 1995.

[38] D. Hammershøi and H. Møller, "Methods for binaural recording and reproduction," *Acta Acustica united with Acustica*, vol. 88, pp. 303–311, 05 2002.

[39] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[40] M. Vorländer, *Sound fields in cavities and in rooms*, pp. 53–68. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[41] M. Noisternig, T. Musil, A. Sontacchi, and R. Holdrich, "3d binaural sound reproduction using a virtual ambisonic approach," in *IEEE International Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS '03. 2003*, pp. 174–178, July 2003.

[42] J. O. Smith, "Physical audio signal processing." `http://ccrma.stanford.edu/~jos/pasp/Spherical_Waves_Point_Source.html`. online book, 2010 edition, accessed 1-July-2019.

[43] Wikibooks, "Engineering acoustics/outdoor sound propagation — wikibooks, the free textbook project." `https://en.wikibooks.org/w/index.php?title=Engineering_Acoustics/Outdoor_Sound_Propagation&oldid=3547399`. [Online; accessed 1-July-2019].

[44] J. O. Smith, "Introduction to digital filters with audio applications." `https://ccrma.stanford.edu/~jos/filters/Minimum_Phase_Polynomials.html`. online book, 2007 edition, accessed 09-August-2019.

[45] M. Vorländer, *Aspects of real-time processing*, pp. 267–278. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.