Rendu sonore dynamique d'environnements complexes

Raphaël Loyet

Université Claude Bernard Lyon 1

Université de Lyon

N ° d'ordre : 304

Année 2012

Thèse de l'université de Lyon

Délivrée par

L'université Claude Bernard Lyon 1

École doctorale InfoMaths

Diplôme de doctorat

(arrêté du 7 août 2006) soutenu publiquement le 18 décembre 2012 par

Raphaël Loyet

Dynamic Sound Rendering of Complex Environments

Rendu sonore dynamique d'environnements complexes

Directeur de thèse : Victor Ostromoukhov

Jury

M. Kadi Bouatouch,	*rapporteur*
M. Nicolas Tsingos,	*rapporteur*
M. Victor Ostromoukhov,	*directeur de thèse*
M. Judicaël Picault,	*président du jury*
M. Jean-Claude Iehl,	*examinateur*
M. Julien Maillard,	*examinateur*

Acknowledgments

Remerciements

S'il ne devait y avoir qu'un remerciement, il irait aux membres de ma famille pour leur soutien indéfectible durant les cinq années de ces travaux et toutes les années qui ont précédé. Un grand grand merci à mes parents pour avoir toujours cru en moi, pour m'avoir toujours soutenu tant personnellement que financièrement pendant mes études. Un grand merci à mon petit frère, pour n'être plus si petit que ça. Et à celle qui fût autrefois ma petite sœur. Celle qui est finalement devenue la grande sœur à la fin de ces travaux.

Mes remerciements vont ensuite à ceux qui ont porté ce travail et ont rendu cette soutenance possible. Bernard Péroche et Jean-Claude Iehl pour m'avoir encadré dès le début de ma thèse et Victor Ostromoukhov pour avoir repris le flambeau lors du départ à la retraite de mon directeur de thèse. Merci aux membres du LIRIS pour leurs nombreux conseils et commentaires concernant ma présentation.

Merci à l'équipe du CSTB pour m'avoir accueilli dans ses locaux pendant les trois premières années de mes travaux. Un merci particulier à Nicolas Noé et Dirk van Maerck pour les discussions que nous avons eues au sujet de ma thèse, pour leur très grande expertise du sujet et pour leurs styles diamétralement opposés. Surtout, merci à Julien Maillard pour avoir supporté dans son bureau un doctorant qui a la mauvaise habitude de faire de la musique (certains diraient du bruit) avec tout ce qui lui passe sous les doigts. Mais aussi pour son soutien constant, pour tous ses éclaircissements dans le domaine du traitement du signal, pour sa grande rigueur qui a amélioré la qualité de mes travaux et de ce manuscrit.

Un merci très chaleureux aux membres du jury venus tous trois spécialement de la côte ouest pour l'occasion (certaines côtes ouest étant plus éloignées que d'autres). Merci à Nicolas Tsingos et Kadi Bouatouch pour avoir accepté de rapporter mes travaux, pour leurs nombreux commentaires et annotations. Merci aussi à Judicael Picault, ce président du jury qui a décortiqué mon manuscrit tel un rapporteur.

J'ai eu la chance pendant ces travaux d'être porté et transporté par des amies, des amantes qui m'ont soutenu à leur façon. Merci Céline pour m'avoir porté et supporté pendant toutes ces années. Merci Eglantine pour m'avoir libéré et émancipé. Merci Fanny pour tes passions, pour m'avoir conduit si haut que l'air se fait rare. Merci Charlotte pour la ligne en pointillés tracée ensemble depuis deux ans, pour notre insoutenable légèreté et pour les quatre premiers chapitres. Merci aux passantes pour n'avoir rien demandé.

Un grand merci à tous les amis, ceux qui ont toujours été là avant, pendant et qui, je le sais resteront encore là pour un bout de route ensemble.

Merci à tous ceux qui m'ont inspiré, en vrac et de façon non exhaustive : Brassens, Brel, Kundera, Brahms, Aronofski, Lebowski, Jardin, Joe Simpson, Cobain, Кустурица, Bregović, Les bérus, Almodovar, Les Têtes Raides, Balanescu Quartet, Leprest, Django, Bratsch, les VRP, Mano Solo, Pierre Schaeffer, the Clash, The Presidents of the USA ...

Merci à tous ceux avec qui nous nous sommes inspirés mutuellement, toujours en vrac : la 5ème Danse Hongroise, Yebarov, les Scared of Bumps, les Patates à la Cave, les Skalators, les Trois Moustiquaires, Das Band, les Laborieux du Dépliant, les Woodchucks, les Shpiritus Movens, les musiciens du café Bayard, tous ceux rencontrés sur la route.

Enfin, merci à tous ces gens, toutes ces images, tous ces sons furtifs, qui rendent la vie si douce : les regards d'Istanbul, les joueurs de cymbalum de Barcelone ou de Freiburg, les compagnons de cordées, Sibiu, le Женекия, la petite bergère qui a perdu son troupeau au fond du canyon de Colca, le désert de sel d'Uyuni, Huayna Potosi, Montserrat, la pluie irlandaise, le feu écossais, le vétérinaire de Gradesnica, la Slovaquie, le Machu Picchu, Eddy, le Mont Aiguille, le Boléro de Ravel et tout ce qui reste à découvrir ...

Raphaël Loyet
27 décembre 2012

Contents

List of Figures

List of Tables

List of Algorithms

List of Notations

A	Surface area
c	Celerity of the sound
C_{80}	Clarity
C_n	Corrective coefficient
$\mathcal{C}$	Clustering function
$\mathcal{C}_S$	Spatial clustering function
$\mathcal{C}_T$	Temporal clustering function
d	Distance
d_R	Distance traveled by a ray
d_{dopp}	Doppler shift
D	Detection function
D_{50}	Definition
D_{max}	Maximal density of rays in a cluster
$\mathcal{E}$	Emitter (sound source)
f	Frequency
f_r	BRDF
$f_{r\mathtt{S}}$	Specular BRDF
$f_{r\mathtt{D}}$	Diffuse BRDF
$F_{i\to j}$	Form factor from i to j
F_s	Sampling frequency
G	Geometric term
g	Geometric term in Temporal Intensity Algebra (TIA) form
h	General impulse response
$h_{\alpha^\mathcal{E}}$	Source directivity filter
$h_{\alpha^\mathcal{R}}$	HRTF filter
$\mathcal{H}^{\alpha_m}$	Sound source convolved with air absorption
$\mathbf{I}$	Intensity
$\mathbf{I}_\mathcal{E}$	Intensity emitted by the sound source
$\mathbf{I}(\theta)$	Irradiance in direction θ
$\Im$	Imaginary part

l	Time dependent radiance
L	Radiance
L_p	Sound pressure level
L_I	Sound intensity level
L_W	Acoustical power
$\mathbf{n}$	Normal to a surface
N_{IS}	Number of image sources
N_P	Number of particles
N_{refl}	Number of reflections
N_{RIR}	Number of coefficients of a RIR
N_{walls}	Number of walls
p	Sound pressure
$\tilde{p}$	Root Mean Square (RMS) sound pressure
P	Particle
r	Radius of a sphere
$r_{\mathcal{R}}$	Radius of the receiver
$r(\mathbf{x}, \mathbf{x}', \mathbf{\Omega})$	Reflection kernel
R	Ray
$\mathcal{R}$	Sound Receiver
$\Re$	Real part
s	General signal
$s_{\mathcal{E}}$	Emitted signal
$s_{\mathtt{E}}$	Emitted signal with directivity
$s_{\mathtt{ES}*}$	Signal after specular reflections
s_i	Input signal
s_o	Output signal
S	Suppressor
$\hat{S}$	Fourier transform of signal s
S_r	Temporal displacement operator
$\tilde{S}_r$	Propagation operator
t	Time

T_{60}	Reverberation time
$\mathcal{T}$	Termination criterion
V	Visibility function
w	Window signal
w_h	Hann window signal
W	Radiant energy
$\mathbf{x}$	Position in 3D space
(x, y, z)	Cartesian coordinates
$(x_\mathcal{E}, y_\mathcal{E}, z_\mathcal{E})$	Cartesian coordinates of a sound source
$(x_\mathcal{I}, y_\mathcal{I}, z_\mathcal{I})$	Cartesian coordinates of an intersection
$(x_\mathcal{R}, y_\mathcal{R}, z_\mathcal{R})$	Cartesian coordinates of a receiver
α	Absorption coefficient
α_m	Absorption of the medium
β	Splitting coefficient
δ	Dirac distribution
γ	Cluster
λ	Wavelength
ϕ_s	Phase of signal s
Φ	Energy
$\Phi_\mathtt{D}$	Energy reflected diffusely
$\Phi_\mathcal{E}$	Emitted Energy
Φ_R	Energy carried by a ray
$\Phi_\mathtt{S}$	Energy reflected specularly
τ_{high}	Temporal source separation threshold
τ_{low}	Temporal echo threshold
(θ, ϕ)	Orientation in polar coordinates
$(\theta_\mathcal{E}, \phi_\mathcal{E})$	Orientation of a sound source in polar coordinates
$(\theta_\mathcal{R}, \phi_\mathcal{R})$	Orientation of a sound receiver in polar coordinates
$\vartheta_\mathtt{S}$	Specular reflectance
$\vartheta_\mathtt{D}$	Diffuse reflectance
ξ	Uniformly distributed random number

List of Acronyms

ADC	Analog-to-Digital Converter
API	Application Programming Interface
BEM	Boundary Element Method
BRDF	Bidirectional Reflectance Distribution Function
BSA	Binaural Spatialization Algorithm
BSSRDF	Bi-directional Subsurface Scattering Reflectance Distribution Function
COLA	Constant-OverLap and Add
CPU	Computer Processing Unit
CSTB	Centre Scientifique et Technique du Bâtiment
DAC	Digital-to-Analog Converter
DAT	Digital Audio Tape
DFT	Discrete Fourier Transform
DRT	Deterministic Ray Tracing
DSP	Digital Signal Processing
ESPRO	*Espace de Projection*
FEM	Finite Element Method
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HRTF	Head Related Transfer Functions
IACC	Interaural Cross-Correlation
ICA	Independent Component Analysis
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response

IR Impulse Response

IRCAM *Institut de Recherche et de Coordination Acoustique/Musique*

ITD Interaural Time Difference

LF Lateral Fraction

LFC Lateral Fraction Cosine

MC Monte Carlo

MCRT Monte Carlo Ray Tracing

MLS Maximum Length Sequence

OL Off-Line

PTB *Physikalisch-Technische Bundesanstalt*

RARE Room Acoustic Rendering Equation

RIR Room Impulse Response

RMS Root Mean Square

RT Real-Time

SCD Specular/Cluster/Diffuse

SRT Stochastic Ray Tracing

TIA Temporal Intensity Algebra

UML Unified Modeling Language

VBAP Vector Base Amplitude Panning

WOLA Weighted-OverLap-Add

Abstract

During the past twenty years many studies have been conducted in the field of auralization, which aims at rendering audible the results of an acoustic simulation. These studies have mainly focused on the propagation algorithms and the sound field audio rendering for complex environments. Currently, much research concentrates on real-time audio rendering.

This thesis addresses the problematic of real-time audio rendering of complex environments according to four axes: sound propagation, Digital Signal Processing (DSP), spatial perception of sound and computational optimizations. In the field of propagation, a method that aims at analyzing the variety of existing algorithms is proposed. This method yields two algorithms dedicated to the real-time propagation of both specular and diffuse information. In the field of DSP, the auralization is performed with an efficient binaural spatialization module for the most significant specular information, and a GPU convolution algorithm for the diffuse sound field auralization. The most significant paths are extracted thanks to a perceptive model based on temporal and spatial masking of the specular contributions. Finally, the implementation of these algorithms on recent computer architectures, taking advantage of the parallel processing of the new CPUs, and the benefits of GPUs for DSP calculations is presented.

Keywords

Auralization – real-time – perceptive reduction – subjective evaluation – sound propagation – ray-tracing – binaural audio rendering – GPU – parallel computing – multi-thread – progressive impulse response update

Résumé

De nombreuses études ont été menées lors des vingt dernières années dans le domaine de l'auralisation. Elles consistent à rendre audible les résultats d'une simulation acoustique. Ces études se sont majoritairement focalisées sur les algorithmes de propagation et la restitution du champ acoustique dans des environnements complexes. Actuellement, de nombreux travaux portent sur le rendu sonore en temps réel.

Cette thèse aborde la problématique du rendu sonore dynamique d'environnements complexes selon quatre axes : la propagation des ondes sonores, le traitement du signal, la perception spatiale du son et l'optimisation informatique. Dans le domaine de la propagation, une méthode permettant d'analyser la variété des algorithmes présents dans la bibliographie est proposée. A partir de cette méthode d'analyse, deux algorithmes dédiés à la restitution en temps réel des champs spéculaires et diffus ont été extraits. Dans le domaine du traitement du signal, la restitution est réalisée à l'aide d'un algorithme optimisé de spatialisation binaurale pour les chemins spéculaires les plus significatifs et un algorithme de convolution sur carte graphique pour la restitution du champ diffus. Les chemins les plus significatifs sont extraits grace à un modèle perceptif basé sur le masquage temporel et spatial des contributions spéculaires. Finalement, l'implémentation de ces algorithmes sur des architectures parallèles récentes en prenant en compte les nouvelles architectures multi-cœurs et les nouvelles cartes graphiques est présenté.

Mots clés

Auralisation – temps réel – réduction perceptive – évaluation subjective – propagation du l'onde sonore – lancer de rayons – restitution binaurale – rendu sur carte graphique – programmation parallèle – mise à jour progressive de la réponse impulsionnelle

Introduction

The first reaction of an acoustician investigating a new room is to clap in his hands, and listen. This empirical method was always used to gain a first insight of the acoustics of the place. The most experienced acoustician or sound engineers maintain that this method will always be very practical to perceive the acoustical qualities of a room. There are two main drawbacks with this empirical method: it is very subjective and it can only be performed in existing buildings. Regarding the first drawback, the discipline of room acoustics provides a certain number of methods and indicators to provide an objective analysis of an enclosed space. These methods are usually based on the analysis of the Room Impulse Response (RIR). The RIR presents the result of the propagation of an impulsive sound between a source and a receiver in an environment. It can be measured with *e.g.* a balloon pop or a gun shoot as impulsive sounds. This is, in some way, similar to a hand clap, but more precise. Here are some examples of questions that are usually addressed by architects to acoustician when designing a room:

- What will be the acoustics of this room?
- What type of materials should I choose to improve the intelligibility of this room?
- What is the influence of this diffuser on the sound perceived in front of the stage?
- How can I increase the reverberation time of this room?

These questions lead us to the second drawback of the hand clapping method; the room does not necessary exist when the previous questions are addressed. Many predictive algorithms were implemented during the past 35 years to estimate the acoustics of virtual spaces. In this document, virtual spaces refer to 3D models that represent existing or non existing places. We especially focus on a family of algorithms based on geometrical acoustic propagation, where the propagation of sound between a source and a receiver is represented by geometrical elements such as rays, beams, particles...

Context

This study is related to the field of virtual reality, and especially spatial audio rendering, *i.e.*, auralization. It covers the evaluation and auralization of the acoustics of an enclosed space, taking into account its geometry and the physical properties of the materials. The audio rendering is interactive: the source and the receiver can move within the virtual scene. These elements allow estimation of the acoustical qualities of a place before its construction or refurbishment through listening tests. It is then possible to estimate and compare different construction scenarios, and thus increase user comfort. The previous studies conducted in the field of real-time auralization are either based on a pre-calculation step [see *e.g.* *Siltanen et al.*, 2009], or on an approximate rendering of the late reverberation [see *e.g.* *Funkhouser et al.*, 2004]. The main strength of these algorithms is their low computation cost, that make them compatible with interactive simulations. However, the approximations of the late reverberation and the pre-calculation time could be prohibitive for both the quality and the calculation time of multiple scenarios of buildings. In parallel, *Hacıhabiboğlu and Murtagh* [2008] proposed a method to reduce the

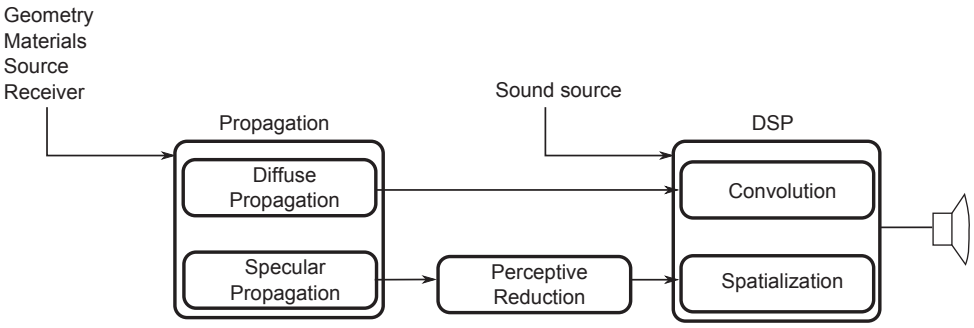


Figure 1: Auralization pipeline.

volume of information perceived by the listener. This reduction method is based on the spatial perception of sounds, and the masking schemes associated. However, this method is computationally expensive and thus not directly relevant for real-time auralization. Figure 1 shows an overview of the auralization pipeline. It includes all the steps from propagation to audio rendering. A review of the previous works will be presented at the beginning of each chapter.

Goal of the dissertation

A review of the aforementioned methods shows that it is possible to realize an interactive navigation in complex environments. It also shows that it is possible to significantly reduce the volume of information without altering the quality of the rendering, *i.e.*, by focusing on the most significant information from a perceptual point of view. However, there is currently no optimal solution permitting the exploitation of the perceptive reduction phenomenon in an auralization algorithm.

This justifies this study, which consists in developing an algorithm to estimate and render audible the acoustics of an enclosed place when the source or the listener are moving.

Contributions of the dissertation

The main contribution of this study is to provide a global method for dynamic sound rendering of complex environments. To reach this goal, improvements over existing approaches were introduced in the fields of sound propagation, Digital Signal Processing (DSP), psycho-acoustics and computational optimization.

In the field of sound propagation, we first present an algorithm based on radiosity and image source to extract exhaustively all acoustical paths between a source and a receiver in a virtual scene. This algorithm is coupled with an analysis method based on the representation of these paths as a grammar or a tree. This analysis yields to a decomposition between pure specular and diffuse sound fields. Starting from these observations, we present a criterion that evaluates the spatial coherence of pure specular and diffuse sound fields for each order of reflection. We also present two algorithms adapted to real-time propagation. The first is based on deterministic ray tracing, it is dedicated to the rendering of

specular paths. The second is based on Monte Carlo Ray Tracing (MCRT) and is dedicated to the rendering of the diffuse sound field. Both have an incremental mechanism to perform the propagation step by step in order to have an incremental auralization system. This incremental rendering process is another contribution of this study. Starting from the analysis of the execution times of all the modules that compose the real-time auralization framework, we present a step by step decomposition for the calculation of the Room Impulse Response (RIR). As we deal with dynamic rendering in this study, either the source or the receiver may move in the virtual scene. Each time one of the elements moves, the first specular and diffuse reflections are recalculated. When the source or the receiver stop moving, further reflections are updated, until the whole Room Impulse Response (RIR) is updated to the new position.

The contribution in the field of Digital Signal Processing (DSP) consists in a rearrangement of the binaural spatialization algorithm presented by *Deille et al.* [2006a]. We analyzed this algorithm and reordered the processing steps to have an efficient real-time auralization of specular contributions. The other Digital Signal Processing (DSP) contribution consists in a convolution algorithm on GPU. This convolution is dedicated to the auralization of the diffuse field.

Finally, the last contribution is an extension of the works conducted by *Hacıhabiboğlu and Murtagh* [2008] on the perceptive reduction of specular information. We present a clustering algorithm that reduces the number of specular paths rendered in order to meet real-time requirements with a high quality level. In previous studies, the auralization of specular paths was generally based on the arrival time of sound paths, or their level. Here, we focus the rendering calculations on the most significant paths from a perceptive point of view, and thus concentrate the calculation on perceptive rather than physical parameters.

Organization of the document

This document is decomposed in six main chapters. Chapter 1 presents the basic materials in the fields of sound propagation, Digital Signal Processing (DSP), psycho-acoustics and room acoustics. These notions are presented to help the reader in the comprehension of the rest of the document. Some of them are extended at the beginning of the dedicated chapters when necessary.

The three next chapters are dedicated to sound propagation, auralization and perceptive reduction respectively. They all share the same structure, with a substantial state of the art in the considered domain, followed by the presentation of our contribution about this domain.

Chapter 2 focuses on the propagation of sound wave in the field of geometrical acoustics. The state of the art consists in a presentation of the major geometrical propagation algorithms such as ray tracing, particle tracing, image sources ... Our contribution is divided in two parts. (i) The presentation of an algorithm based on image sources and radiosity used to generate exhaustively all diffuse and specular paths in a virtual scene. With this algorithm, we propose two analysis methods of the propagation algorithms based on grammar and tree representations. The implementation of this algorithm yields a criterion used to establish a hierarchy between paths for real-time rendering. This hierarchy is based on the spatial coherence of the specular and diffuse sound fields at various orders of reflection. (ii) Starting from these observations, we present at the end of Chapter 2 two algorithms suited for real-time

propagation of specular and diffuse paths.

Chapter 3 presents the DSP methods used to auralize the pressure sound field. We first present the methods from the bibliography used for early and late reverberation. Then we present one real-time algorithm dealing with specular field audio rendering and another real-time algorithm dedicated to diffuse field auralization. An original aspect of this thesis is to decompose the rendering into diffuse and specular paths instead of early and late reflections. Specular reflections are rendered using a binaural rendering algorithm based on the Independent Component Analysis (ICA) decomposition of the Head Related Transfer Functions (HRTF) [adapted from *Deille et al.*, 2006a]. The diffuse field auralization is performed with convolution techniques in the frequency domain. We present the approach to convert data between the propagation and the auralization stages. We also present the implementation of this module on a Graphical Processing Unit (GPU).

Chapter 4 deals with the spatial perception of sound. The state of the art presents the different aspects of spatial hearing, including the localization of a sound source in 3D space, the localization of multiple sources and the masking that occurs between coherent sources. In this chapter, we present a perceptive clustering algorithm based on the works conducted by *Hacıhabiboğlu and Murtagh* [2008]. This algorithm applies a perceptive reduction on the specular paths of the simulations. A subjective study presents the tests that were conducted to provide appropriate parameters to tune the perceptive algorithm. This chapter concludes with a discussion on the implementation and the benefits of such a mechanism in an auralization framework.

Chapter 5 is a more technical chapter discussing the implementation of the algorithms presented in the three previous chapters on recent computer architectures. It addresses the issues of both multi thread implementation on CPU and heterogeneous development, across both CPU and GPU. This chapter finally presents the progressive impulse response update that aims at performing the audio rendering in real-time while the modules that compose the auralization framework do not necessarily reach real-time requirements. This method explains the decomposition of the calculation in different steps. The most significant parts of the process are updated when the source or the receiver move in the virtual scene. Then, the rest of the Room Impulse Response (RIR) is gradually updated.

Finally, Chapter 6 presents the validation of our algorithms. Two aspects are studied: the validity of the RIR generated according to the standard objective parameters of room acoustics as well as the execution time our algorithms (to justify real-time rendering). The tests were performed on the scenes of the third round robin in room acoustics [*Bork*, 2005b].

1

Representations of sound

This chapter presents the basic materials about sound in the field of physics (see Section 1.1), signal processing (see Section 1.2), psycho-acoustics (see Section 1.3) and room acoustics (see Section 1.4). These notions will be developed when necessary in the following chapters.

Contents

What is a sound? The answers given to this simple question can be looked upon from three different points of view. In this chapter, we present the three different definitions of sound that will be developed in the following chapters. The first is explained from the point of view of the physician. We present the different physical quantities used to describe sound waves, and to characterize acoustical sources and receivers. We then explain the concept of virtual scenes and its geometric definition. In Section 1.1.5 we present the Room Acoustic Rendering Equation (RARE) as formulated by *Siltanen et al.* [2007] and *Kiminki* [2005]. In the following section, we describe the sound from Digital Signal Processing (DSP) perspective. We present the basic notions of signal processing, and in particular DSP. The time/frequency analysis is presented in order to introduce DSP operators such as filtering, delay, convolution and Doppler shifting and to analyze their complexity. We conclude this chapter with the point of view of the psycho-acoustician. We present how the sound is perceived by the human ear, and how it is possible to locate a sound in a three dimensional space. In Section 1.3 we present the methods used to simulate a virtual sound in three dimensions. Finally, we present some basic concepts of room acoustics. They are used as a bridge between subjective and objective ways of evaluation audio simulations. Objective parameters are used in Chapter 5 to validate our model.

1.1 The physics of sound propagation

We present in this section the basic notions of physics commonly used to study the propagation of sound. In common language, the sound represents the set of all phenomena that can be perceived by the human ear, *i.e.*, that can be heard. Listening is possible thanks to the eardrum, that is located in the middle ear. The eardrum is sensitive to the fluctuations of pressure in the air.

In an enclosed space, it is possible to observe that after a short period without turbulences the pressure reaches a steady state. That means a state with a local energy minimum. The propagation of the wave occurs when particles are moving around their rest state. It depends on the characteristics of the medium. In the air, the speed of propagation can be approximated by[1]

$$c = 343.2\sqrt{\frac{273.15 + \upsilon}{293.15}} \tag{1.1}$$

Where c is called the speed of sound in the air in $[\mathrm{m} \cdot \mathrm{s}^{-1}]$, and υ is the temperature in degrees Celsius.

The motion of the wave in the air can be represented by a mass spring system *Cadoz et al.* [1993]. When one mass is displaced by an excitation (the membrane of a speaker, the vibration of a string ...) it takes kinetic energy. The attached spring is then compressed, stores potential energy and transfers it to the neighboring masses, and so on.

As the sound travels in the air, or as it interacts with materials, it gets attenuated. This attenuation is dependent on the frequency.

[1] A more precise definition can be found in *ISO9613-3* [1996].

1.1.1 The physics of sound

Sound wave propagation can be described with some assumptions on the wavelength relative to the dimensions of the surrounding objects. By analogy to light propagation, the wave front propagation of the sound is generally represented by rays, beams or particles. This type of representation is developed in the field of geometrical acoustics. We present here the basic quantities used to describe acoustical phenomena under the assumptions of linear acoustics. Note that for clarity reasons, the frequency parameter is omitted in the following equations.

Pressure

While listening to sounds, the ear is sensitive to the variations of pressure around a rest state p_0. While the medium is compressed and decompressed, the total pressure p_{tot} at the measurement point is dependent on space and time. The sound pressure p can be expressed as

$$p = p_{tot} - p_0 \tag{1.2}$$

These are very small pressure fluctuations that are perceived by the ear. In equation 1.2 the scales are very different between the values. The static pressure p_0 is $1,013 \cdot 10^5$ Pa or one bar in normal laboratory conditions (20 ° C, no wind, ...), whereas the acoustical pressure p is very rarely above ten Pa. The RMS pressure (p_{RMS} or $\tilde{p}$), the value that we can read on our measurement tools (or that our ear perceives) is defined during a time period T:

$$\tilde{p} = p_{RMS} = \sqrt{\frac{1}{T}\int_0^T p^2(t)\mathrm{d}t} \tag{1.3}$$

Flux

The flux, also called power or radiant power is the rate of flow of energy. It is in general expressed in watts [W] or in joules per second [$\mathrm{J} \cdot \mathrm{s}^{-1}$]. Therefore, if W is the radiant energy, the radiant power Φ is defined by

$$\Phi = \frac{dW}{dt} \tag{1.4}$$

Irradiance and Intensity

Irradiance $\mathbf{I}(\theta)$ is the power radiated per surface area, it is expressed in watt per meter square [$\mathrm{W} \cdot \mathrm{m}^{-2}$]:

$$\mathbf{I}(\theta) = \frac{d\Phi}{dA}\cos\theta \tag{1.5}$$

where θ is the angle between the normal $\mathbf{n}$ to the surface, and the measured intensity. $d\Phi$ the power leaving the surface dA. According to *Morfey* [2000], the intensity $\mathbf{I}$ is the measure of the irradiance in

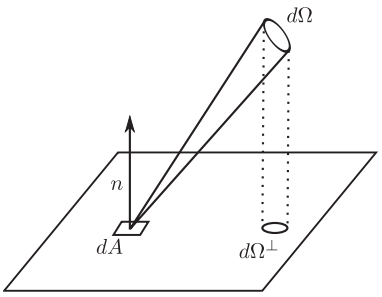


Figure 1.1: Radiance notations: Ω is a solid angle, and $\Omega^{\perp}$ the projected solid angle.

orthogonal incidence angle, $\mathbf{I}$ is expressed as

$$\mathbf{I} = \frac{d\Phi}{dA} \tag{1.6}$$

In the case of plane wave propagation, the intensity can be linked to the pressure by the following formula:

$$|\mathbf{I}| = \frac{\tilde{p}^2}{\rho_0 c} \tag{1.7}$$

where $\rho_0 c$ is the characteristic impedance of the medium in $[\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}]$[2].

Radiance

Radiance L is the power per projected solid angle per unit area[3]. It is measured in watts per steradian per meter square $[\mathrm{W} \cdot \mathrm{st}^{-1} \cdot \mathrm{m}^{-2}]$. Radiance is a function of position and orientation, therefore a 5D function[4]. It is defined as (the notations of solid angle and projected solid angle refers to Figure 1.1):

$$L = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\Omega^{\perp}\mathrm{d}A} = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\Omega\cos\theta\mathrm{d}A} \tag{1.8}$$

We deduce from equations 1.6 and 1.8 that

$$L = \frac{\mathrm{d}\mathbf{I}}{\mathrm{d}\Omega\cos\theta} \tag{1.9}$$

With these last two expressions, it is important to note that:

- The radiance measured from an object does not depend on the distance.
- If an object emits twice as much energy, but covers twice the solid angle of another object, the total radiance received will be the same.

[2] The characteristic impedance for air at normal conditions, 20 °C is 414 $\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$.
[3] Details about solid angle geometry can be found in Appendix A.1.
[4] Three dimensions for space and two dimensions for orientation.

- Since the solid angle is connected to the square of the distance, we have the well known property that the energy received from objects diminishes with the square of the distance.

It is also important to note that radiance is invariant along a straight line — the radiance going from point A to point B is equal to the radiance incoming at point A from the direction of point B.

1.1.2 What is a virtual scene ?

Since the invention of computer graphics, it is possible to construct representations of geometrical entities on the screen of a computer. The first investigations in this field were conducted by *Bresenham* [1965] who provided algorithms to find an easy way to discretize a line on a computer screen — where the discretization steps are the pixels of the screen. With the modernization of computers, it is now possible to have 3D representations of very large and complex models. It is for example possible to visualize a representation of a full city with a model that has a precision of the order of the centimeter. It is also possible to have dynamic displacement inside this model in order to provide the user a virtual visit.

In this study, we restrict our field to the auralization of complex enclosed spaces such as concert halls, theaters, classrooms, or recording studios. But, the major advantage with the concept of virtual scenes, is that they can model nonexistent projects, like architect ideas at the design stage, archaeological reconstructions of no longer existing buildings, or unrealizable buildings for the purpose of artistic experimentations.

Geometry

The geometry of virtual scenes can be defined by a set of triangles defining walls and other objects of the scene. Triangles are defined by three points in $\mathbb{R}^3$ space, and a normal vector $\mathbf{n}$ that gives the orientation of the triangle. In the following, it is assumed that triangles are not connected by any topological information.

The level of detail for an acoustic virtual scene does not need to be as accurate as the models for visualization, as the effects involved are proportional to the wavelength λ of the effect we want to model.

$$\lambda = \frac{c}{f} \tag{1.10}$$

where f is the frequency in [Hz]. As human ear can detect sounds of frequency f between 20 Hz and 20 kHz[5], the wavelength we are interested in are within the range $[17 \cdot 10^{-3} \text{ m} \rightarrow 17 \text{ m}]$. In practice, the geometries used for acoustics usually have details of the order of the meter [*Siltanen et al.* [2008]]. Scenes with a too high level of details can lead to wrong estimations in low frequencies. This under-sampling of the scenes is a great benefit for acoustics, as the scenes we usually deal which have generally three to four orders of magnitude less triangles than scenes in computer graphics.

[5] See Section 1.3 for more information on psycho-acoustics.

1.1.3 Sound source

Sound sources (or sound emitters) are all the elements of everyday life capable of producing sounds. Some examples of emitters are: musical instruments, people talking, singing, clapping their hands, a door clapping, a hammer knocking on a nail, a bee flying around a head, a drill on a construction site, the wind on the leaves ... In fact, all elements of every day life are capable of producing a sound.

Depending on the geometry of the source, we can define the attenuation pattern $\alpha_{\mathcal{E}}$ as a function of the direction of emission (θ, ϕ in polar coordinates), and the frequency f.

From a geometrical point of view, the sound emitter can be seen as a punctual element of the virtual scene defined by its position in Cartesian coordinates $(x_{\mathcal{E}}, y_{\mathcal{E}}, z_{\mathcal{E}})$ and its orientation in polar coordinates $(\theta_{\mathcal{E}}, \phi_{\mathcal{E}})$. As this study is about dynamic sound rendering, the source can be moved inside the environment to position $(x_{\mathcal{E}}, y_{\mathcal{E}}, z_{\mathcal{E}}) + \Delta(x, y, z)$ or toward orientation $(\theta_{\mathcal{E}}, \phi_{\mathcal{E}}) + \Delta(\theta, \phi)$. In the following, we assume that the free field radiation of the source can be described by its radiation pattern, $\alpha_{\mathcal{E}}$, and the emission signal $s_{\mathcal{E}}(t)$.

From the physical point of view, the elemental source is represented as an infinitesimal pulsing sphere called a monopole. As the monopole radiates spherical sound waves, it sets the particles of the medium into motion. Usually, the sound source is characterized in acoustics by its sound power. The total radiated sound power can be calculated by integrating the sound intensity over a surrounding measurement surface. Thus, for a monopole source (uniform directivity), we have the relation between sound intensity $\mathbf{I}_{\mathcal{E}}$ and sound power $\Phi_{\mathcal{E}}$ at a distance r in free field [after *Vorländer*, 2008]:

$$\mathbf{I}_{\mathcal{E}} = \frac{\Phi_{\mathcal{E}}}{4\pi r^2} \tag{1.11}$$

1.1.4 Sound receiver

A sound receiver can also be seen as a punctual entity in the virtual scene. It is defined by its position in Cartesian coordinates $(x_{\mathcal{R}}, y_{\mathcal{R}}, z_{\mathcal{R}})$ and its orientation in polar coordinates $(\theta_{\mathcal{R}}, \phi_{\mathcal{R}})$ and its directivity $\alpha_{\mathcal{R}}$. Different types of receivers can be defined for virtual reality applications. Usually, the receivers directivity and arrangements have the characteristics of microphones used for studio recording. The characteristics of the microphones can be modeled — directional, cardioid, hyper cardioid, omni-directional, and the way the microphones are placed (see *Mercier et al.* [2006] for more information on microphone arrangement) can be:

- Single microphone,
- XY couple,
- ORTF couple,
- AB couple,
- Microphone ramp,

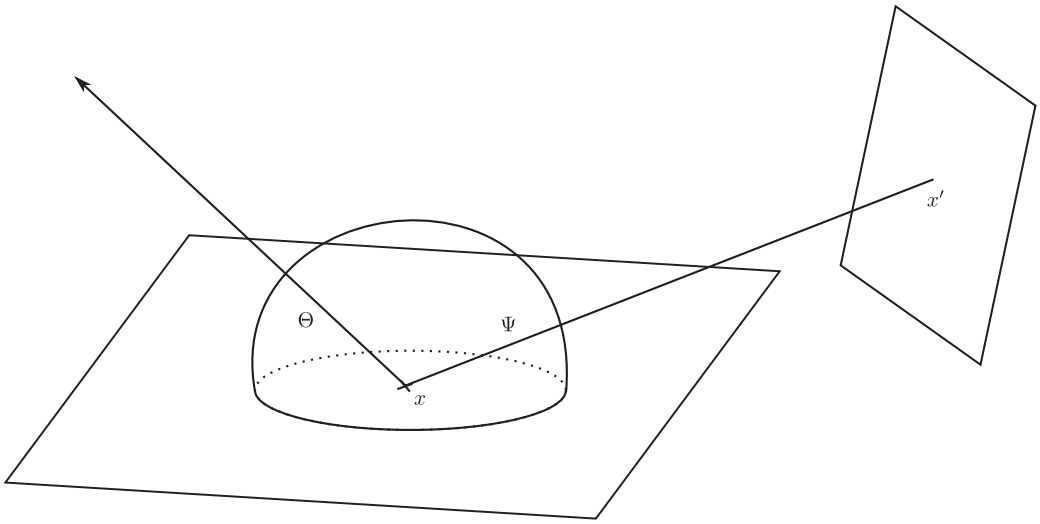


Figure 1.2: Rendering equation notations (*cf.* Equation 1.12).

- "Sound field" Ambisonic,
- Artificial head.

Artificial head recording, uses two microphones positioned at the entrance of the ear canal of a fake torso and head, that aims at recording the sound as human ears could perceive it. The main challenge to record such a sound, is to take into account all reflection and diffraction effects that occur between the sound wave and the listener's body. As the sound reaches the body of the listener, complex interactions occur with the torso, the shoulders, the face, the pinna, and the outer ear canal. In order to obtain a more accurate recording of the sound, and to improve their localization, reproduction of human ear pinna are installed on the dummy head.

Most of the virtual reality simulations try to model the way a human listener perceives the sound in a 3D environment. The head of the receiver is modeled, like a dummy head for studio recording. As the sound reaches the ear it is attenuated depending on the direction of arrival on the head. The attenuation patterns are called Head Related Transfer Functions (HRTF)[6].

1.1.5 Room Acoustic Rendering Equation (RARE)

The propagation of the sound in enclosed spaces can be studied either with analytic methods for very simple geometries or with numerical methods for more complex ones. Numerical methods are generally decomposed in two main categories: wave base methods such as Boundary Element Method (BEM), Finite Element Method (FEM) or radiosity and geometric methods such as rays, particles, beams ... In this section, we present a method used for the analysis of geometrical methods. For an introduction

[6] HRTF are a function of direction of arrival and frequency.

on analytic and wave based methods, the reader may refer to *Vorländer* [2008]. The Room Acoustic Rendering Equation (RARE) is a recent method presented by *Kiminki* [2005] and *Siltanen et al.* [2007] to analyze geometrical sound propagation algorithms. The following equation represents the propagation of radiance in a virtual scene. The analysis is derived from Kajiya's rendering equation [*Kajiya*, 1986] in the field of computer graphics, or its more recent formulation [*Dutre et al.*, 2002] :

$$L(\mathbf{x} \to \mathbf{\Theta}) = L_e(\mathbf{x} \to \mathbf{\Theta}) + \int_{\mathcal{G}} f_r(\mathbf{x}, \mathbf{\Psi} \to \mathbf{\Theta}) L(\mathbf{x}' \to -\mathbf{\Psi}) V(\mathbf{x}, \mathbf{x}') G(\mathbf{x}, \mathbf{x}') \mathrm{d}\mathbf{x}' \tag{1.12}$$

where

- $\to$ means "in the direction of". For instance $(\mathbf{x} \to \mathbf{\Theta})$ means "leaving point $\mathbf{x}$ in direction $\mathbf{\Theta}$, and $(\mathbf{x}, \mathbf{\Psi} \to \mathbf{\Theta})$ means "coming to point $\mathbf{x}$ from direction $\mathbf{\Psi}$ and leaving in direction $\mathbf{\Theta}$.
- $L(\mathbf{x} \to \mathbf{\Theta})$ is the total radiance leaving current patch at point $\mathbf{x}$ in direction $\mathbf{\Theta}$.
- $L_e(\mathbf{x} \to \mathbf{\Theta})$ is the radiance emitted by current patch at point $\mathbf{x}$ in direction $\mathbf{\Theta}$, L_e is zero if current patch is not a sound source.
- $f_r(\mathbf{x}, \mathbf{\Psi} \to \mathbf{\Theta})$ is the BRDF[7].
- $L(\mathbf{x}' \to -\mathbf{\Psi})$ is the radiance at point $\mathbf{x}'$ in direction $-\mathbf{\Psi}$, the direction toward $\mathbf{x}$.
- $V(\mathbf{x}, \mathbf{x}')$ is the visibility function, it is a boolean function that as a value one if the two points $\mathbf{x}$ and $\mathbf{x}'$ see each others, zero otherwise.
- $G(\mathbf{x}, \mathbf{x}')$ is a geometric term (see Figure 1.3).
- $\mathcal{G}$ belongs to $\mathbb{R}^3$, it represents all points on the surfaces of the scene.
- $\int_{\mathcal{G}} \mathrm{d}\mathbf{x}'$ represents the integration on all the surfaces of the scene.

Figure 1.2 presents the notations used in Equation 1.12.

Geometric term G

The geometric term G is defined as

$$G(\mathbf{x}, \mathbf{x}') = \left(\mathbf{n}(\mathbf{x}) \cdot \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|}\right) \left(\mathbf{n}(\mathbf{x}') \cdot \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|}\right) \frac{1}{|\mathbf{x} - \mathbf{x}'|^2} \tag{1.13}$$

where $\mathbf{n}(\mathbf{x})$ is the normal of the patch that contains $\mathbf{x}$. G represents both the geometric divergence that is proportional to the square of the distance between $\mathbf{x}$ and $\mathbf{x}'$, and the non-orthogonality of the

[7] see Section 2.1.3 for more information on BRDF.

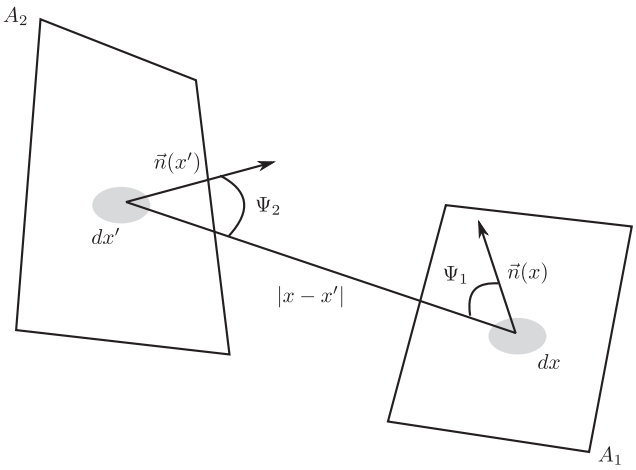


Figure 1.3: Geometric term G notations.

exchange between the two patches. If Ψ_1 is the angle between the vector $\mathbf{x} - \mathbf{x}'$ and $\mathbf{n}(\mathbf{x})$, and Ψ_2 is the angle between the vector $\mathbf{x}' - \mathbf{x}$ and $\mathbf{n}(\mathbf{x}')$ (see Figure 1.3), G can be reformulated as:

$$G(\mathbf{x}, \mathbf{x}') = \frac{\cos \Psi_1 \cos \Psi_2}{|\mathbf{x} - \mathbf{x}'|^2} \tag{1.14}$$

It is important to note that G is reciprocal, *i.e.*, $G(\mathbf{x}, \mathbf{x}') = G(\mathbf{x}', \mathbf{x})$.

Temporal Intensity Algebra (TIA)

In this section, we give the basic results formulated by *Kiminki* [2005] and *Siltanen et al.* [2007]. For more information on the temporal extensions of *Kajiya* [1986] rendering equation, the reader may refer to *Kiminki* [2005]. The first element of the Temporal Intensity Algebra (TIA) is the propagation operator. It is defined as

$$\tilde{S}_r \mathbf{I}(t) = e^{-\alpha_m r} S_r \mathbf{I}(t) = e^{-\alpha_m r} \mathbf{I}\left(t - \frac{r}{c}\right) \tag{1.15}$$

where c is the speed of sound, α_m the absorption coefficient of the medium, $\mathbf{I}$ the intensity (*cf.* Section 1.1.1). S_r is an operator that applies the temporal displacement to $\mathbf{I}$ based on the distance r traveled by the sound. $\tilde{S}_r$ is another operator that applies both the temporal displacement, and the absorption of the medium to $\mathbf{I}$. In the special case of the propagation of an impulsive sound located at $t = 0$, *i.e.*, a Dirac distribution $\delta(t)$, the following property holds:

$$\tilde{S}_r[\mu_1 \delta(t) + \mu_2 \delta(t)] = [\mu_1 \tilde{S}_r + \mu_2 \tilde{S}_r]\delta(t) \tag{1.16}$$

By differentiation the propagation operator can also be applied to irradiance and radiance (*cf.* Section 1.1.1). The second property, additivity is expressed as:

$$\tilde{S}_{r_1}\tilde{S}_{r_2}\delta(t) = \tilde{S}_{r_1+r_2}\delta(t) \tag{1.17}$$

The two previous equations justify that the operators can be combined in the case of specular reflection; the intensity at a point can be computed as the sum of the direct path, and all reflections reaching that point. Using Equation 1.15 with a Dirac distribution, we obtain:

$$\tilde{S}_r\delta(t) = e^{-\alpha_m tc}S_r\delta(t) \tag{1.18}$$

If we incorporate the intensity in the previous definition, we have:

$$\mathbf{I}(t) = \left[\sum_{i=0}^{n_{refl}}(1-\alpha_i)S_{r_i}\delta(t)\right] * e^{-\alpha_m tc}\mathbf{I}_{\mathcal{E}}(t) = \mathcal{H}^{\alpha_m}\sum_{i=0}^{n_{refl}}(1-\alpha_i)S_{r_i}\delta(t) \tag{1.19}$$

where $\mathbf{I}_{\mathcal{E}}(t)$ is the emission intensity varied over time, α_i are the Sabine absorption coefficient of the i^{th} wall intersected, $*$ is the convolution operator. $\mathcal{H}^{\alpha_m}$ is an operator that represents the convolution of a source signal filtered by a medium characterized by its absorption coefficient, α_m.

Geometric term extension g

As we have seen in the previous section, it is necessary to include the time in the rendering equation for acoustics. *Siltanen et al.* [2007] include the propagation operator S_r in the geometric term G which becomes g in lower case for acoustic modeling. g gathers the geometric term G defined in Equation 1.13 and the propagation operator S_r (*cf.* Equation 1.15):

$$g(\mathbf{x},\mathbf{x}') = S_{|\mathbf{x}-\mathbf{x}'|}G(\mathbf{x},\mathbf{x}') = \left(\mathbf{n}(\mathbf{x})\cdot\frac{\mathbf{x}-\mathbf{x}'}{|\mathbf{x}-\mathbf{x}'|}\right)\left(\mathbf{n}(\mathbf{x}')\cdot\frac{\mathbf{x}'-\mathbf{x}}{|\mathbf{x}'-\mathbf{x}|}\right)\frac{S_{|\mathbf{x}-\mathbf{x}'|}}{|\mathbf{x}-\mathbf{x}'|^2} \tag{1.20}$$

Reflection kernel r

In order to simplify the notations of the RARE, the visibility function V, the BRDF f_r and the geometric operator g are gathered into the reflection kernel r:

$$r(\mathbf{x},\mathbf{x}',\Omega) = V(\mathbf{x},\mathbf{x}')f_r(\frac{\mathbf{x}-\mathbf{x}'}{|\mathbf{x}-\mathbf{x}'|},\Omega\rightarrow\mathbf{x}')g(\mathbf{x},\mathbf{x}') \tag{1.21}$$

It is important to note that the time dependence is included in the geometric term g. In order to simplify the writing of equations, time is also omitted in the parameters.

Room Acoustic Rendering Equation (RARE)

Since we consider time-dependent radiance in acoustics, the Room Acoustic Rendering Equation (RARE) has to be defined using the reflection kernel r presented above. In order to differentiate time-dependent from time-independent radiance, the lower case l symbol is used. As for the previous functions, the time parameter is omitted in the definition of the function:

$$l(\mathbf{x} \to \Omega) = l_0(\mathbf{x} \to \Omega) + \int_{\mathcal{G}} r(\mathbf{x}', \mathbf{x}, \Omega) l \left(\mathbf{x}' \to \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|} \right) \mathrm{d}\mathbf{x}' \tag{1.22}$$

where

- $l(\mathbf{x} \to \Omega)$ is the time-dependent radiance leaving point $\mathbf{x}$ in direction Ω,
- $l_0(\mathbf{x} \to \Omega)$ is the emitted radiance (zero if the patch does not belong to a sound source),
- $\int_{\mathcal{G}} \mathrm{d}\mathbf{x}$ is the integral on all the surface points of the scene
- $r(\mathbf{x}, \mathbf{x}', \Omega)$ is the reflection kernel,
- and $l \left(\mathbf{x} \to \frac{\mathbf{x}-\mathbf{x}'}{|\mathbf{x}-\mathbf{x}'|} \right)$ is the incoming, time-dependent radiance in direction $\mathbf{x} - \mathbf{x}'$.

Equation 1.22 represents the radiative exchanges of radiance in the case of room acoustics propagation, the reflection kernel r defined above allows to have a formulation very close to Kajiya's rendering equation (*cf.* Equation 1.12).

With this formulation, it is possible to use the Neumann series[8] solutions presented in the original article of *Kajiya* [1986] on Equation 1.22:

$$l_{i+1}(\mathbf{x} \to \Omega) = \int_{\mathcal{G}} r(\mathbf{x}, \mathbf{x}', \Omega) l_i \left(\mathbf{x} \to \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|} \right) \mathrm{d}\mathbf{x}', \tag{1.23}$$

$$l(\mathbf{x}' \to \Omega) = \sum_{i=0}^{N_{ref}} l_i(\mathbf{x}' \to \Omega). \tag{1.24}$$

This equation shows that the detection of the total radiance leaving one point in the scene can be computed as the the sum of the radiance calculated at each reflection order, and that reflections at each order can be calculated incrementally from the results calculated at previous reflection orders. The objective of the propagation algorithms presented in Sections 2.1.5 and 2.1.6 is to solve this equation.

Emission

In the previous equations, l_0 represents the radiance emitted by a surface. In the case of a point source, l_0 can be taken as the first reflection of the radiance on the walls of the virtual scene. If the source —

[8] See *e.g.* *Kiminki* [2005] of an introduction on Neumann series, and *e.g.* *Guenther and Lee* [1996] for an extended presentation.

the emitter — is located at position $\mathbf{x}_\mathcal{E}$, and the source radiates energy with a pattern $\alpha_\mathcal{E}(\Omega)$, then the time-dependent irradiance from the source to any point $\mathbf{x}$ of the scene is

$$l_0(\mathbf{x} \rightarrow \Omega) = r\left(\mathbf{x}_\mathcal{E}, \mathbf{x}, \Omega\right) V(\mathbf{x}_\mathcal{E}, \mathbf{x}) g_0(\mathbf{x}_\mathcal{E}, \mathbf{x}) \alpha_\mathcal{E}\left(\frac{\mathbf{x} - \mathbf{x}_\mathcal{E}}{|\mathbf{x} - \mathbf{x}_\mathcal{E}|}\right) \tag{1.25}$$

with g_0 the extension of the geometric term (*cf.* Equation 1.20) for the exchange between a punctual entity (the sound source) and a patch of the scene. g_0 is expressed:

$$g_0(\mathbf{x}_\mathcal{E}, \mathbf{x}) = \frac{S_{|\mathbf{x}_\mathcal{E} - \mathbf{x}|}}{4\pi |\mathbf{x}_\mathcal{E} - \mathbf{x}|^2} \lfloor \mathbf{n}(\mathbf{x}) \cdot \frac{\mathbf{x}_\mathcal{E} - \mathbf{x}}{|\mathbf{x}_\mathcal{E} - \mathbf{x}|} \rfloor \tag{1.26}$$

In Equations 1.25 and 1.26,

- $r\left(\mathbf{x}_\mathcal{E}, \mathbf{x}, \Omega\right)$ is the reflection kernel (*cf.* Equation 1.21) for a sound coming from $\mathbf{x}_\mathcal{E}$ to point $\mathbf{x}$ and leaving in direction Ω.
- $V(\mathbf{x}_\mathcal{E}, \mathbf{x})$ is one if the source sees point s, zero otherwise.
- $\alpha_\mathcal{E}\left(\frac{\mathbf{x} - \mathbf{x}_\mathcal{E}}{|\mathbf{x} - \mathbf{x}_\mathcal{E}|}\right)$ is the emission pattern attached to the source.
- $S_{|\mathbf{x}_\mathcal{E} - \mathbf{x}|}$ is the propagation operator between the source and point $\mathbf{x}$
- $\lfloor a \rfloor$ is defined by *Siltanen et al.* [2007] as

$$\lfloor a \rfloor = \begin{cases} a, & a > 0 \\ 0, & a \leq 0 \end{cases}$$

Detection

A punctual receiver can be seen as an infinitesimally small sphere located at $\mathbf{x}_\mathcal{R}$. We consider that the detection of the incident energy is always performed at point $\mathbf{x}_\mathcal{R}$ orthogonally to the surface of this sphere. The receiver can be characterized by a direction dependent transfer function[9] $\alpha_\mathcal{R}(\Omega, \mathbf{I})$. This transfer function gives the attenuation of the incoming intensity $\mathbf{I}$ for a given direction Ω.

In the case of a punctual emitting source, the detection of the direct sound has to be handled separately, so the total detection function, $D(t)$, is the sum of the detection of the direct sound, $D_{direct}(t)$, and the reflected radiance on all the surfaces of the scene, $D_{reflected}(t)$:

$$D(t) = D_{direct}(t) + D_{reflected}(t) \tag{1.27}$$

[9] Sections 1.1.4 and 1.3.4 give more details about these transfer functions, and in particular HRTF.

where

$$D_{direct}(t) = \alpha_{\mathcal{R}} \left(\frac{\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}}{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}|}, \mathcal{H}^{\alpha_m} V(\mathbf{x}_{\mathcal{E}}, \mathbf{x}_{\mathcal{R}}) \frac{S_{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}|}}{4\pi |\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}|^2} \alpha_{\mathcal{E}} \left(\frac{\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}}{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}_{\mathcal{E}}|} \right) \right) \tag{1.28}$$

and

$$D_{reflected}(t) = \int_{\mathcal{G}} \alpha_{\mathcal{R}} \left(\frac{\mathbf{x}_{\mathcal{R}} - \mathbf{x}}{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}|}, \mathcal{H}^{\alpha_m} V(\mathbf{x}, \mathbf{x}_{\mathcal{R}}) \frac{S_{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}|}}{4\pi |\mathbf{x}_{\mathcal{R}} - \mathbf{x}|^2} l \left(\mathbf{x} \to \frac{\mathbf{x}_{\mathcal{R}} - \mathbf{x}}{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}|} \right) \lfloor \mathbf{n}(\mathbf{x}) \cdot \frac{\mathbf{x}_{\mathcal{R}} - \mathbf{x}}{|\mathbf{x}_{\mathcal{R}} - \mathbf{x}|} \rfloor \right) \mathrm{d}\mathbf{x} \tag{1.29}$$

1.2 Digital signal processing for auralization

In virtual acoustics, the signals involved in the simulations are digital signals with broadband content. Unlike single harmonic signals, also referred to as pure-tone signals, broadband signals contain multiple frequencies. A pure tone signal s at frequency f_0, with initial phase ϕ_0 is expressed by

$$s(t) = \sin(2\pi f_0 t + \phi_0) \tag{1.30}$$

Every signal, either harmonic or inharmonic, can be represented by a sum of pure tone signals. For inharmonic sounds, the sum is infinite.This is the basis of Fourier analysis defined in the next section.

When sound travels in a virtual scene, each path is associated with the sound trajectory. It can be modeled as a combination of filtering and delay operators applied to the signal representation of sound pressure. Different methods exist to process delay and filtering, both in time and frequency domains. These methods are presented in Sections 1.2.6 and 1.2.7. The set of all paths reaching a listener can also be modeled by a single entity called Impulse Response (IR)[10]. Convolution is a method that is generally used to apply the acoustic response of a room to an anechoic sound (*cf.* Section 1.2.5).

Finally, as we deal with dynamic sound rendering, the effect of time varying delays, *i.e.*, *Doppler effect*, has to be studied. *Doppler effect* describes the frequential modifications of a sound when the source and the receiver are moving. This is described in Section 1.2.8

More information on signal processing can be found in dedicated books like *Soize* [1993]; *Smith* [1997]; *Kahrs and Brandenburg* [1998]; *Tanguy* [2007] or *Tisserand et al.* [2008].

1.2.1 Analog *vs.* Digital signal processing

A signal can be studied either with analog or digital formulation. As expected, analog signal processing considers analog, *i.e.*, continuous signals, whereas digital signal processing considers digital, *i.e.*, discrete signals. The aim of this study is to provide computational tools for real-time auralization. The signals

[10] Impulse responses can also be recorded in existing rooms.

studied are digital. The analog formulation gives a good theoretical framework for the study of signals, and can sometimes simplify the formulation of signal processing problems. On the other side, the digital formulation gives methods that can be implemented directly in algorithms. The latter formulation was chosen for this study, unless the continuous formulation proves more convenient.

In auralization, the link between analog signals and digital signals is the Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) of the audio interface. The first converts the input analog signal from the microphone, whereas the second converts the digital output signal to the speakers. One important characteristic of ADC is that the sampling rate of the digital signal must respect the Shanon-Nyquist principle: the signal cannot contain frequencies above the Shanon-Nyquist frequency which is defined as half the sampling frequency. For example, a signal sampled at 48kHz cannot contain frequencies above 24kHz. More information on sampling theories can be found in signal processing books like *Smith* [1997].

Different notations are used depending on the type of signal used: Continuous temporal signals are generally noted $s(t)$, with, t, the time ($t \in \mathbb{R}$). Digital signals are noted $s[n]$, where n is the sample number ($n \in \mathbb{Z}$).

1.2.2 Fourier analysis

Most of the auralization systems do not process pure tone signals, as most sounds surrounding us are complex sounds. In order to propagate sounds in an auralization system, the sound source needs to be recorded without reverberation. Such sound is called an *anechoic sound*. It can be recorded in an *anechoic room* with an absorption coefficient on all the walls very close to one for all frequencies. Fourier analysis is a very important tool in signal processing. Assuming signals represent linear phenomena, a given signal, $s(t)$, can be expressed in the frequency domain by its *Fourier transform*, $\hat{S}(f)$:

$$\hat{S}(f) = \int_{-\infty}^{+\infty} s(t)\ e^{-i2\pi ft}\ \mathrm{d}t \tag{1.31}$$

$\hat{S}(f)$ is a complex valued function where f denotes the frequency in $[Hz]$. We have the symmetrical expression, called *inverse Fourier transform* that gives a signal in the temporal domain, starting from its expression in frequency domain, $s(t)$:

$$s(t) = \int_{-\infty}^{+\infty} \hat{S}(f)\ e^{i2\pi ft}\ \mathrm{d}f \tag{1.32}$$

As $\hat{S}(f)$ belongs to $\mathbb{C}$, two important functions are defined to analyze the signal in the frequency domain. The first is the magnitude of the signal $|\hat{S}|(f)$, it defines the frequency content of the signal[11]:

$$|\hat{S}(f)| = \sqrt{\Re(\hat{S}(f))^2 + \Im(\hat{S}(f))^2} \tag{1.33}$$

[11] $\Re(\hat{S})$ and $\Im(\hat{S})$ represents respectively the real and imaginary parts of the complex signal $\hat{S}$, *i.e.*, $\hat{S} = \Re(\hat{S}) + i\Im(\hat{S})$.

The second important information is the phase of the signal:

$$\phi_{\hat{S}}(f) = \tan^{-1}\left(\frac{\Im(\hat{S}(f))}{\Re(\hat{S}(f))}\right) \tag{1.34}$$

Discrete Fourier Transform (DFT)

All formulas defined above for continuous signals have an equivalent for discrete signals. For signals of size N, the DFT is defined as

$$\hat{S}[f] = \sum_{n=0}^{N-1} s[n]\ e^{-i2\pi fn/N} \tag{1.35}$$

The Inverse Discrete Fourier Transform (IDFT) is defined as

$$s[n] = \sum_{f=0}^{N-1} \hat{S}[f]\ e^{i2\pi fn/N} \tag{1.36}$$

Some faster methods were developed to speed up the calculation of DFT. The most famous is the Fast Fourier Transform (FFT), its implementation is well described by *Smith* [1997]. Section 3.3.2 presents some works on FFT implementations on GPU.

1.2.3 Impulse response

One of the most used functions in signal processing is the Dirac distribution (noted δ). In digital systems, it is a vector filled with zeros, except for the first value that is set to one. More details about delta distribution can be found in Appendix A.3. An arbitrary signal can be seen as the sum of delayed δ weighted by the amplitude of the sample. When a Dirac distribution is filtered through a linear system, the resulting output is referred to as the *impulse response* of the system. Two linear systems that have the same impulse response are said to be identical. Sometimes, the DSP systems are also characterized by their *transfer function*. The *transfer function* is defined as the Laplace Transform of the impulse response [*Smith*, 1997]. For causal systems, this is equivalent to the DFT of the impulse response.

One of the most used impulse responses in room acoustics is the Room Impulse Response (RIR). It describes the reverberation of a room excited by an impulsive sound — empiric measurement were usually performed using a gun shot or a balloon pop. According to *Müller and Massarani* [2001], impulse responses are more accurately measured with sweep signals or pseudorandom noise.

1.2.4 Echogram

Another signal of interest in room acoustics simulations is the echogram. It is defined as the magnitude (see Equation 1.33) of the impulse response represented on a logarithmic scale (see Section 1.3.1). The echogram is used for instance to represent the response of a room with energetic propagation algorithms

such as particle tracing (see Section 2.1.6). As the echogram represents the propagation of the energy, it is proportional to the square of the pressure: therefore, the echogram does not contain phase information.

The echograms that are typically used in virtual acoustics simulations are integrated frequency dependent echograms *i.e.*, discrete echogram with a temporal step of Δt. The frequency dependent echograms are used for instance to estimate the energy carried by a particle traveling a virtual scene in a given frequency band, *e.g.* , octave or third octave bands. The integrated echograms are widely used in room acoustics. For example, they can be used for the calculation of the objective room acoustics parameters defined in Section 1.4. They can also be used to control the precision of propagation algorithms such as Monte Carlo path tracing (see Section 2.5.2).

1.2.5 Convolution

Convolution is the operator which implements the filtering of an input signal by an impulse response. In room acoustics, the response of the room can be seen as a linear system. The convolution of an anechoic sound by the impulse response of a room will produce a sound with the reverberation of the room. For an input signal, s_i, an impulse response, h, and an output signal, s_o, the convolution notation is

$$s_o(t) = s_i(t) * h(t) \tag{1.37}$$

As we have seen previously, if the input signal of a linear system modeled by its impulse response is a delta function, the output will be its impulse response:

$$h(t) = \delta(t) * h(t) \tag{1.38}$$

Convolution is a linear operator, *i.e.*, it respects homogeneity and additivity.

One important property about this operator, is that a convolution in the time domain is equivalent to a multiplication in the frequency domain:

$$s_o(t) = s_i(t) * h(t) \qquad \underrightarrow{DFT} \qquad \hat{S}_o(f) = \hat{S}_i(f) \cdot \hat{H}(f) \tag{1.39}$$

Due to the symmetry of the Fourier transform, the opposite relation is true: a convolution in the frequency domain is equivalent to a multiplication in the time domain, *i.e.*,

$$\hat{S}_o(f) = \hat{S}_i(f) * \hat{H}(f) \qquad \underrightarrow{IDFT} \qquad s_o(t) = s_i(t) \cdot h(t) \tag{1.40}$$

For digital signals, the convolution operator in the time domain is defined as

$$s_o[n] = \sum_{j=0}^{N_h - 1} h[j] s_i[n - j] \tag{1.41}$$

with N_h the number of samples of the impulse response. If N_S is the number of samples of the input sequence, $s_i[n]$, the output sequence, $s_o[n]$ will be $N_{s_o} = N_h + N_{s_i} - 1$ sample long.

In the frequency domain it is directly obtained with

$$\hat{S}_o[f] = \hat{H}[f]\hat{S}_i[f] \tag{1.42}$$

From the two previous equations, we see that time domain convolution requires N_h multiplications per output samples, whereas convolution in the frequency domain only requires one. Note that to completely analyze the complexity of both algorithms, the complexity of FFT and Inverse Fast Fourier Transform (IFFT) algorithms have to be estimated [see *Oppenheim and Schafer*, 1975].

1.2.6 Filtering

Filtering is an important operation in the dynamic sound rendering process, as several steps of the propagation are implemented as filtering operations in the auralization. For example, the direction dependent emission pattern of a sound source, $\alpha_{\mathcal{E}}$, and the direction dependent detection function, $\alpha_{\mathcal{R}}$, of a receiver (see Section 1.1.5) correspond to filtering operations. The frequency dependent intersection of a ray with a wall of the virtual scene represents also a filtering operation.

There are two main families of filters: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters.

Finite Impulse Response (FIR) filtering

FIR filtering is simply defined as the convolution of an input signal with the *filter kernel* — also called the *convolution kernel*. The response of the filter can be calculated either in the frequency or in the time domain depending on the applications. Efficient convolution with long FIR filters is usually performed in the frequency domain (see Section 1.2.5).

Infinite Impulse Response (IIR) filtering

IIR filters, also referred to as recursive filters, provide approximations of the impulse response of FIR filters with a lower number of coefficients, *i.e.*, a lower filter order. The synthesis of IIR filters can be made using the *z-transform*[12]. *Deille et al.* [2006a] uses for instance IIR filters to implement an optimized version of HRTF filtering for real-time auralization.

The temporal formulation of a recursive filter is:

$$s_o[n] = \frac{1}{a_0}\left(b_0 s_i[n] + b_1 s_i[n-1] + \ldots - a_1 s_i[n-1] - a_2 s_i[n-2] + \ldots\right) \tag{1.43}$$

where a_i and b_i represent the polynomial coefficients of the denominator and the numerator of the filter response respectively. The roots of the denominator and numerator are called the poles and the zeros of the filter.

[12] More information on Chebyshev and Butterworth filter synthesis can be found in *e.g.* *Smith* [1997].

Octave band filtering

FIR and IIR filters enable the implementation of arbitrary filter responses. Octave band filtering refers to specific filter bands for modeling responses with constant magnitude within octave bands. The approach is to discretize the frequency in frequency bands. Generally, octave or third octave bands are used, but other types of discretization like Bark bands [*Kahrs and Brandenburg*, 1998] based on perception can be used. We present in this section the filtering process for octave bands, as it is the one used in this study, but other discretization may be used with the same method[13]. The principle of octave band filtering is depicted in Figure 3.7. An input signal is decomposed (filtered) in several octave bands. The signal remains in the time domain, but only a part of the frequency content is kept. Then, an attenuation gain is applied to each signal, this corresponds to the gain of the center frequency of the octave band. Once the gains of all frequency bands have been applied, the signals are summed to recompose the broadband signal.

The main advantage of this method is its simplicity, and its low computational cost. We have used in our implementation IIR filters to decompose the original signal, but, for some applications, FIR filters implemented with FFT can be used to reduce computational load[14]. Note that the octave band filtering does not allow representation of the phase information of the transfer function to be modeled.

* * *

Filtering is a linear operation, this is an important property for auralization, as it permits to change the order of the filtering operations, and to factorize some parts of the processing. This property is intensively used in Chapter 3.

1.2.7 Delay

Delay is another essential operation in DSP, it represents a temporal displacement of the signal. This is for instance the operation used to model the propagation time of a sound ray in room acoustics. Delay was a very complex operation to perform on analog circuits as it requires the use of circuits with memory. With a computer, the implementation of a delay becomes really simple: It consists in placing the samples in memory for the time of their delay. The computational structure is generally called a *delay line*. The delay can be represented by the convolution of a Dirac distribution delayed by t_d:

$$s_o[n] = \delta[n - t_d] * s_i[n] = s_i[n - t_d] \tag{1.44}$$

Fourier transform of the delay is:

$$\hat{S}_o[f] = \hat{S}_i[f] e^{i2\pi t - t_d} \tag{1.45}$$

The problem with the previous formulation of the delay is that t_d must be a positive integer, as the signal is causal and discrete. For non integer values, fractional delay techniques based on interpolation

[13] Only the filters properties and number changes.
[14] Details about FFT bank techniques can be found in *e.g.* *Smith* [2009].

must be used. Here is the formulation of a fractional delay based on linear interpolation:

$$s_o[n] = r_d \; s_i[n - n_d] + (1 - r_d) \; s[n - n_d - 1] \tag{1.46}$$

n_d represents the integer part of t_d, and r_d the rest[15]. This formula can be extended with higher order of interpolation methods as described in Appendix A.5.

Delay, is also a linear operation, for instance, the delay of ray traveling in a virtual scene is the sum of the delay between each intersection.

Fixed size delay line The implementation of a delay line is in general memory consuming. A delay is simply a vector of floating point values that store the delayed signal for a certain time. In virtual acoustics, the time a signal is stored is less or equal the maximal time of the RIR we want to generate — *e.g.* for a three seconds RIR sampled at 44.1 kHz, a maximum of 132.300 samples have to be stored.

Delay pointers Another approach to implement delay lines is to read the delayed samples directly from the original signal loaded in memory. With this method, only one pointer per delay line is necessarily, thus, reducing the associated memory consumption to a minimum. The major problem with this technique is that the delay has to be the first operation applied to the signal, which is not necessary the best choice in virtual reality applications.

* * *

We present in Section 3.2 a modified version of the delay lines that applies the delays right after the octave band filtering of the signal.

1.2.8 Doppler shift

The Doppler effect is a well known effect in acoustics, it is a modification of the frequency induced by the relative displacement of a source and a receiver. This effect explains for instance why the sound of a car driving at constant speed has higher frequency when the car approaches the listener than when it moves away. The frequency perceived by a listener will be modified by a factor

$$d_{dopp} = \frac{1 - \frac{(\mathbf{x}_\mathcal{E} - \mathbf{x}_\mathcal{R})}{||\mathbf{x}_\mathcal{E} - \mathbf{x}_\mathcal{R}||} \frac{\mathbf{v}_\mathcal{R}}{c}}{1 - \frac{(\mathbf{x}_\mathcal{E} - \mathbf{x}_\mathcal{R})}{||\mathbf{x}_\mathcal{E} - \mathbf{x}_\mathcal{R}||} \frac{\mathbf{v}_\mathcal{E}}{c}} \tag{1.47}$$

where $\mathbf{x}_\mathcal{E}$ and $\mathbf{x}_\mathcal{R}$ are the positions of the source and receiver respectively, and $\mathbf{v}_\mathcal{E}$ and $\mathbf{v}_\mathcal{R}$, their velocity.

The Doppler shift can be implemented in the time domain using the fractional delay presented in Equation 1.46. The delay time is interpolated between the previous and the current positions of the source and the receiver.

Note that the Doppler shifting is no-longer a linear operation, so it cannot be interchanged with other DSP operations such as delay or filtering.

[15] *e.g.* if $t_d = 54.8$, then $n_d = 54$ and $r_d = 0.8$.

1.2.9 Block processing for real-time algorithms

Real-time processing implies that the processing time of one sample of the discrete output signal must be inferior to the sampling period of the signal[16], assuming a single channel output signal. If the processing time does not meet this requirement, sound artifacts occur, and the simulation fails. Most of the real-time systems do not process the signal on a sample basis, they are usually organized as blocks of samples. At every clock signal, the sound card requests to the processing unit the last processed sample block. Most of the time, the blocks have a size that is a power of two, samples are well suited to the binary structure of the computer memory,[17] as well as frequency domain processing based on FFT.

Overlap methods

When the processing applied to the signals evolves with time, two strategies can be adapted to process the blocks of signal. Special attention must be paid to the first and last elements of the block. This is where there is a high probability to have discontinuities during the processing. The first strategy is to apply a sample based processing, such as interpolated gains or fractional delays between two values. With this type of process, the increment must be independent of the block size. In the case of interpolated gains, the process applies the gain to current processing sample, then increments the gain for the next processing sample. If the next sample belongs to another block, this makes no difference, thus a smooth transition is performed between blocks.

The second method is suited to processing that applies to blocks of signal, such as FFT filtering, with a non constant filter in time. In this case, the sample per sample method is no-longer valid and discontinuities will occur at the block junctions. The method used to have a smooth transition between block is called the *Overlap method*. This method creates a smooth interpolation between the blocks, and thus suppresses the audible artifacts caused by the discontinuity of the filter. We have for instance implemented overlap methods for the convolution of an anechoic sound, with a RIR evolving in time (see Section 5.4). More information about overlap methods can be found in *e.g. Smith* [2009].

1.3 Spatial hearing – The perception of sound

The end goal of auralization algorithms is to produce sounds. The resulting sound samples are presented to listeners. The topic of this research is to produce sound samples that represent as accurately as possible the reverberation of enclosed spaces. Psychoacoustics is the science of the sound perception. We present in this chapter some basic notions of psychoacoustics that will be developed later in Chapter 4. In our study, psychoacoustics is used both to validate and to optimize our algorithms. As our study is focused on the perception of reverberation, a great emphasis is put on spatial hearing characteristics. Thus, in Section 1.3.3, we present the mechanism of 3D sound to locate sound in space and its representation using Head Related Transfer Functions (HRTF). Then, in Section 1.3.4 we present various

[16] For a signal sampled at 44100 Hz, the processing time of a sample must be under 22.6 μs.

[17] In our implementation, most of the blocks have a size of $2^9 = 512$, this implies that a block of signal must be processed in less than 11.6 ms. The choice of 512 as the block size is motivated by perceptive parameters presented in Chapter 4.

rendering techniques used to produce 3D sounds.

The information gathered in this chapter are mainly extracted from *Blauert* [1999]; *Vorländer* [2008]; *Tsingos* [1998]; *Emerit* [1995] and *Nicol et al.* [2008].

1.3.1 Perceived intensity of sound

We have presented in Section 1.1.1 different physical characteristics of sound, including sound intensity, pressure and power. The human ear is generally sensitive to variations of intensity from 10^{-12} W $\cdot$ m^{-2} (audition threshold) to 1 W $\cdot$ m^{-2} (pain threshold). These limit values vary generally with many parameters including the age of the listener, the *health* of his ear or the frequency content of the sound.

As the variations of intensity perceived by human ear cover a wide range of values, of around twelve orders of magnitude, the linear representation of the intensity is not well adapted to the perception of sounds. So, it is common in acoustics to use a logarithmic scale to represent the sound intensity levels. The intensity level, L_I, is expressed in *decibels* as

$$L_I = 10 \log_{10} \left(\frac{I}{I_0} \right) \tag{1.48}$$

where I_0 is the hearing threshold, $I_0 = 10^{-12}$ W $\cdot$ m^{-2}. Generally the intensity level is expressed in [dB IL] (decibel intensity level).

Similar formulations exist for sound pressure level, L_p, and acoustical power, L_W, that are respectively expressed in [dB SPL] (decibel sound pressure level) and [dB PL] (decibel power level):

$$L_p = 20 \log_{10} \left(\frac{\tilde{p}}{\tilde{p_0}} \right) \tag{1.49}$$

$$L_W = 10 \log_{10} \left(\frac{\Phi}{\Phi_0} \right) \tag{1.50}$$

with the reference values (audition threshold), $\tilde{p_0} = 2.10^{-5}$ N.m^{-2} and $\Phi_0 = 10^{-12}$ W.

1.3.2 Frequency perception of sound

As for the perception of intensity, the human ear is able to distinguish a wide variety of frequencies in sounds. For a children's ear, it is possible to perceive frequencies in the 20 Hz to 20 kHz range. The upper limit falls in general to a value around 15 kHz for an adult. The upper value of the frequency perceived by human ear explains for instance the choice of the standard sampling frequencies for digital signals. We have seen in Section 1.2.1 that the frequency content of an analog signal to be convected must respect the Shanon-Nyquist condition, *i.e.*, the signal must not have frequency content above half the sampling frequency. The sampling frequencies commonly used in digital recording are 44.1 kHz for the compact disk, or 48 kHz for professional standards, like the Digital Audio Tape (DAT).

The perception of frequency content is unequal across the spectrum. The ear sensitivity reaches its highest value for frequencies in the range 1 to 3 kHz. These frequencies correspond to the spectrum of

the spoken voice.

Different models were developed for the analysis of the frequency content of sound. Each one of them has pros and cons depending on the application. In this section, we present briefly four frequency analysis models, and discuss their use in the context of room acoustics auralization.

Fine bandwidth model The fine band model considers the full spectrum of the signal, *i.e.*, all frequency components of the signal Fourier transform evenly spaced between zero and the Shannon-Nyquist frequency $F_s/2$ where F_s is the sampling frequency. In this model, each component is complex including both amplitude and phase information. Fine band models are important for effects such as frequency interferences. In our study, fine bands are used for the spatialization of sounds using HRTF (see Sections 1.3.3 and 3.2). The drawback of this method is that the processing of all frequencies using fine band methods is generally more expensive than the following three methods.

Octave bandwidth model In the octave band model, the frequency spectrum is divided into logarithmically spaced frequency bands where each center frequency is obtained by a factor of two. With this method, the spectral representation of the signal is a rough approximation of the original signal. It is generally used for signals that have slow and smooth variation in frequency space. Material attenuation is a good example of the usage of octave bands for acoustical modeling. Most of the databases of frequencial attenuations of the materials (often referred as Sabine attenuation coefficients) are presented in octave bands. In this model, the first and last bands are extended to zero and $F_s/2$ respectively. In terms of Digital Signal Processing (DSP) operations, this corresponds to using pass band filters for the central bands, a low-pass filter for the first band, and a high-pass filter for the last band. One example of material characterization with octave bands is presented by *Bork* [2005a] where the materials are presented for six octave bands between 125 Hz and 4 kHz. In our implementation, we used the octave band decomposition for the propagation of sound in the virtual scene. This method is used to model the intersections with the walls of the virtual scene, and air absorption (see Section 3.2.1).

Third octave bandwidth model Another method commonly used in room acoustic simulations is the third octave band decomposition. It is very close to the octave band decomposition, except that third octave band decomposition is closer to the frequency resolution of the human ear. *Tsingos* [1998] uses for instance a decomposition twenty-four third octave bands for his propagation algorithms. This decomposition is also interesting for the frequency masking study. A sound with a high intensity in a given third octave band is able to mask sounds in the adjacent bands. As we have not studied the frequency masking of sounds in our study, and in order to speed up the processing of the algorithms, we did not use third octave band filtering.

Bark bandwidth model Finally, another decomposition method commonly used in psycho-acoustics is Bark bands. This decomposition is no longer based on a mathematical decomposition of the frequency spectrum. Instead, it is based on perceptive parameters. The frequency bands for which the human

ear is more sensitive will be narrower, *i.e.*, will contain less frequencies, and thus will be more precise. *Blauert* [1999] describes the decomposition of a signal using Bark bands.

Note that for the above three wide frequency band models, the phase information is discarded since strong variations are usually present within each band.

1.3.3 Spatial perception of sound

The perception of sound in 3D space involves a number of cognitive mechanisms. It can be characterized as a function of three parameters $(d_{\mathcal{E}}, \theta_{\mathcal{E}}, \phi_{\mathcal{E}})$, which define the position of the sound source (or emitter), $\mathcal{E}$ relative to the head in polar coordinates.

$d_{\mathcal{E}}$ represents the distance between the source and the receiver. The perception of the distance is first related to a decrease of the intensity as the source moves (a decrease of six decibels occurs when the distance source/receiver is doubled). The second effect linked to the distance is the frequency dependent attenuation due to air absorption. Air acts as a complex low-pass filter on sounds, and, the longer a sound travels, the more it gets attenuated. A model of air attenuation can be found in *ISO9613-3* [1996]. This is the model implemented in our simulations. Also in reverberant environments such as enclosed spaces, the perception of the distance is related to the ratio of direct sound (perceived as a single source)to reverberated sound (perceived as a spatially diffuse source).

To locate a sound source in azimuth, $\theta_{\mathcal{E}}$, and in elevation, $\phi_{\mathcal{E}}$, two mechanisms are involved. First, the difference of arrival time at the two ears. This is called Interaural Time Difference (ITD). This information gives a first approximation of the position of the sound in space. The ITD is very short compared to the distance traveled by the sound. It can be zero if the sound is located in the plane of azimuth $\theta_{\mathcal{E}} = 0,^{\circ}$, and never exceeds 1 ms in the case of a sound coming from the side of the listener $(\theta_{\mathcal{E}} = \pm 90,^{\circ})$.

The localization in azimuth and elevation in 3D is also due to the complex interactions of the sound with the body of the listener. As the sound reaches the listener, complex modifications of the sound occur. The sound gets diffracted and attenuated many times before it reaches the eardrum. Depending on their direction of arrival, and their frequency content, it can interact with the torso, the shoulders, the face and the different parts of the ear of the listener. These modifications applied to the sound can be modeled as complex transfer functions called Head Related Transfer Functions (HRTF). The problem with HRTF is that they are unique for every person. Thus, to be accurate, HRTF have to be measured and adapted to every listener. Much work on HRTF has been conducted to find a model which is a good representation of the averaged ear characteristics. In this study, we started from works conducted by *Emerit et al.* [1995]; *Emerit* [1995] and *Deille et al.* [2006a].

* * *

Reverberation is another effect linked to sound in 3D space. The reverberation is created by the multiple reflections of sound on the walls of an enclosed space. The purpose of this work is to provide an accurate simulation of the reverberation in enclosed spaces. So, the concept of reverberation is presented in the following chapters. Chapter 2 presents the algorithms implemented to find the different

reflections of a sound in 3D space and to implement the reverberation model. Chapter 3 presents the DSP operations applied to the path to auralize the sound paths generated by propagation algorithms. Chapter 4 is dedicated to the perceptive analysis of the reverberation. In our simulation, only the most significant paths, from a perceptive point of view are kept. Finally, Chapter 5 presents the methods used to update interactively the reverberation in virtual reality applications.

1.3.4 3D audio rendering techniques

Starting from the description of how a listener perceives a sound in 3D space, we present briefly in this section the methods used for the 3D auralization of sound.

Monophonic This very basic type of auralization is used when the system is limited to a single speaker. Monophonic rendering is not strictly speaking a method of spatial rendering as it is necessary to have at least two signals to produce a spatialized sound.

Stereophonie and Vector Base Amplitude Panning (VBAP) VBAP is a method used for the auralization with a set of speakers placed in 2D or 3D, for rendering in the azimuthal plan, and in 3D space respectively. In this method, the same signal is sent to two or three of the speakers with appropriate gains. The choice of the speakers and the gains depend on the position of the virtual source. This technique has he advantage to be easy to implement, to fit irregular arrangements of speakers (as long as their position is known). On the other hand, some artifacts may occur during the auralization process. They are due to the fact that the speakers radiates the same signal at different positions in space, thus, the signal reaching the listener's ears may be incoherent. In the simple case of two speakers located in front of the listener, this technique is called stereophony. This was the first technique used for sound spatialization. More information on VBAP techniques can be found in *Pulkki* [1997] and *Pulkki et al.* [1999].

Binaural This method is dedicated to headphone auralization. It improves the rendering by taking into account Head Related Transfer Functions (HRTF). The basic implementation for binaural spatialization takes each path provided by the propagation algorithm, delay them according to the Interaural Time Difference (ITD) and filter them with the HRTF. The details about binaural rendering are widely described in Section 3.1.3. Similar methods exist with a broadcasting over speakers instead of headphones. This technique is usually called *binaural broadcasting over speakers* or *trans-auralization*. The implementation over speakers is a bit more complex, as the signal emitted by the speaker opposite to the listener's ear has to be canceled using cross talk cancellation filtering. More information on transaural techniques can be found in *e.g. Sibbald* [2009].

Ambisonics Ambisonic broadcasting can be seen as an extension of VBAP methods. It is a method for recording and auralization of 3D sound using speakers. As for VBAP, ambisonic systems can be built for 2D and 3D reproduction. The basic principle of ambisonics is to recreate a sound field around the

listener using the signal emitted by the speakers. In order to produce such a sound field, gains and delays are applied to the signal emitted by the speakers, in order to create interfering sound waves. In VBAP techniques, only the two or three speakers located in the direction of the virtual sound source are active. With Ambisonic systems, more speakers may interact in order to produce the correct interferences. The interested reader may refer to *Daniel* [2001].

1.4 Room Acoustics – Structure of the reverberation

Room acoustics provides useful methods to estimate the quality of a room. This section explains how these tools can be used to provide further analysis of our algorithms, as well as to estimate the real need for enclosed space audio simulation.

1.4.1 Room Impulse Response (RIR) analysis

The Room Impulse Response (RIR) represents the temporal distribution of the sound contributions between a source and a receiver in a room. The RIR is an important signal, as its convolution with an anechoic sound produces a sound reverberated with the characteristics of the room. Two main families of methods may be used to obtain a RIR. It is first possible to measure the response, using an impulsive sound, *i.e.*, a gun shot or a balloon pop in an existing room. More accurate methods, such as Maximum Length Sequence (MLS) or sweep methods (see *e.g. Müller and Massarani* [2001]) are often used for RIR measurements. More information on RIR measure can be found in room acoustics books such as *Kuttruff* [1973]. The other method used to create RIR is numerical simulation. We present in Chapter 2 many algorithms that can be used to model a RIR.

Most of the time, the RIR are decomposed into two or three parts according to the arrival time of the contributions. Some of the decompositions that are often seen in room acoustics publications are :

Early/Late reflections — The first contributions of the sound are considered as early reflections, they have to be modeled precisely in order to catch the characteristics of the room. The late reflections, often called *diffuse field*, can be modeled with less accuracy.

Early/Middle/Late reflections This decomposition is similar to the previous one, except that the portion of the RIR where the early and late reflections overlap is given a special attention. Nearly all commercial room acoustics software use this model.

Direct sound Some authors also extract other parts of the RIR depending on the characteristics of the algorithms. Most of the time, the direct sound, *i.e.*, the first contribution between the source and the receiver, is processed apart from the rest of the RIR.

* * *

In Section 3.5, we propose a new way to decompose the Room Impulse Response (RIR). We called this decomposition Specular/Cluster/Diffuse (SCD) model. This decomposition is linked to the algorithms used to produce the virtual sound field and psycho-acoustic parameters. In this model, the most significant reflections (that are not necessarily the first ones) are treated independently from the rest of the Room Impulse Response (RIR).

1.4.2 Reverberation time

The reverberation time is defined as the duration over which the sound intensity decreases by a factor 10^{-6}, equivalent to a gain of -60 dB (T_{60}).

Reverberation time can be estimated from geometric simulation. Sabine provided a hypothesis based on empirical results that gives a good approximation of T_{60} [after *Jouhaneau*, 2000]:

$$T_{60} = \frac{0.16V}{A + 4mV} \tag{1.51}$$

With V, the volume of the room, $4mV$, the atmospheric absorption proportional to room volume, and A, Sabine's equivalent absorbent area:

$$A = \sum_i A_i = \sum_i S_i \alpha_i \tag{1.52}$$

With A_i, the equivalent absorption area of wall, S_i, the associated surface area, α_i Sabine's energy absorption coefficient per unit surface.

To remain valid, the calculation of T_{60} must respect Sabine's hypothesis:

- Homogeneous diffusion
- Room neither too small nor too absorbent ($\alpha_i << 1$)
- Measure have to be made in far field

Sabine's absorption coefficients are in general given in octave bands for different materials.

T_{60} can also be estimated from the decay curve associated with the RIR. The formulation of the decay curve, h^2, is

$$h^2(t) = \int_t^\infty p^2(\tau) \mathrm{d}\tau \tag{1.53}$$

with p the impulse response.

The reverberation time, is generally evaluated from a logarithmic slope regression between -5 dB and -35 dB of the decay curve. This evaluation multiplied by 2 gives the reverberation time, it is generally called T_{30}.

The two previous formulations of the reverberation time are important, as they are theoretically very close. They provide a simple way to validate the algorithms as a first approximation.

1.4.3 Objective parameters for the evaluation of a room

As subjective evaluation of room impulse response is a long process, many objective indicators have been introduced in room acoustics in order to have a simple way to compare the acoustic quality of a room. We present here the two main parameters that we used for the validation of our algorithms, but the reader may refer to *e.g.* *Vorländer* [2008]; *Kuttruff* [1973]; *Bork* [2005a] or *Beranek* [2003] for more objective parameters.

Definition D_{50} The definition is the percentage of energy reaching the listener during the 50 first milliseconds over total energy. It helps to characterize the percentage of intelligibility of speech. This information is essential for classroom or amphitheater modeling:

$$D_{50} = 100 \left(\frac{\int_0^{0.05} p^2(t)\mathrm{d}t}{\int_0^{+\infty} p^2(t)\mathrm{d}t} \right) \tag{1.54}$$

Clarity C_{80} The clarity expressed in dB is based on a ratio of energy during the first 80 ms over the energy after 80 ms.

$$C_{80} = 10 \log \left(\frac{\int_0^{0.08} p^2(t)\mathrm{d}t}{\int_{0.08}^{+\infty} p^2(t)\mathrm{d}t} \right) \tag{1.55}$$

* * *

We presented in this chapter the basic materials in the fields of physics, signal processing, psychoacoustics and room acoustics to understand the following chapters. Some important notions were discussed briefly in this chapter, they will be developed when necessary in the following chapters.

2

Analysis and implementation of propagation algorithms

*In this chapter, we present the propagation of sound in enclosed spaces. We begin with a state of the art (*cf. *Section 2.1) of the various existing techniques. Starting from these observations, we introduce an algorithm that aims at enumerating all paths between a source and a receiver in a virtual scene (*cf. *Section 2.3). The objective of this algorithm is to analyze the propagation algorithms in terms of propagation depth, and the types of interactions they model. The algorithms can be classified thanks to the grammar presented in Section 2.1.7. As a result of this analysis, we present a criterion to optimize the rendering hierarchy of real-time algorithms: the spatial coherence criterion (see Section 2.4). Finally, the implementation of two optimized algorithms that fit the problem of real-time auralization is presented. The first one is used to reproduce pure specular paths (*cf. *Section 2.5.1). The second one is dedicated to the auralization of diffuse sound field (*cf. *Section 2.5.2.)*

Contents

2.1 State of the art

Acoustic propagation of a sound in a room necessarily involves the description of the sound wave propagation. Sound is emitted from a sound source in every direction of the virtual scene. The frequency content of the sound is modified as it interacts with objects of the scene. A receiver located in the scene gathers the sound information every time it is intersected by a sound wave. Propagation algorithms have been widely studied since the invention of the computer, and a wide variety of algorithms have been developed to take into account the different characteristics of propagation. None of these algorithms are perfect, and this section gives an analysis of the most popular algorithms in order to find the one that best fits dynamic sound rendering in complex environments.

This section only deals with geometrical acoustic algorithms. They are better suited for auralization, as described in Chapter 3. *Vorländer* [2008] gives an introduction to other methods called *wave based methods* — BEM, FEM, difference methods ... —. Another family of methods is based on statistical models of the reverberation. They are independent of the geometry and thus do not fit for an accurate rendering of complex environments. Some of these methods, based on recursive filters are presented by *Kahrs and Brandenburg* [1998].

It was decided not to take into consideration diffraction in our analysis. In order to be complete, our model should add this missing information, but in many cases, diffraction has only a minor influence on the auralization of the calculations in an enclosed space. This restriction leads to the point that coupled volumes, or volumes with occluders will be misestimated.

In our analysis, we have split the auralization algorithms into two categories depending on the intersection types they model. The two major intersections modeled in geometric algorithms are specular reflections and diffusion. We first talk about the basic algorithms that only deal with specular reflections, such as image source (*cf.* Section 2.1.5) or Deterministic Ray Tracing (DRT) (*cf.* Section 2.1.5). We then describe the radiosity algorithm (*cf.* Section 2.1.6) that is well suited for pure diffuse propagation, and conclude with various algorithms that combine both specular and diffuse reflections such as MCRT.

2.1.1 Ray or particle propagation

This chapter is restricted to geometric propagation algorithms; these algorithms are in general described as algorithms of the family of ray tracing. Before we present the details of such algorithms, it is important to describe what is a ray, and how it travels in the virtual scene. In the definition of some algorithms, the concept of ray is sometimes replaced by particle, or other entities like beam, frustum or cone. The naming of these elements has evolved over the years, and we propose in this section a naming convention depending on the entity that is propagated.

The first ideas of ray-based algorithms that were developed are the Stochastic Ray Tracing (SRT) algorithms by *Allred and Newhouse* [1958a,b]. *Kulowski* [1985] later presented a detailed article on the implementation of ray tracing algorithms on computer architectures. The basic stochastic algorithm is based on the generation of sound rays in every direction, starting from the sound source. The rays are

then reflected specularly[1]. Then the receiving sphere collects the particles as it is intersected by a ray. In this method, a histogram is filled, and all rays crossing the receiver during a short period of time are summed.

Another approach of ray tracing called Deterministic Ray Tracing (DRT) is briefly discussed in *Vorländer* [2008]. The method is similar to the previous one, except that the collection is no-longer based on the summation of rays, but on the detection of a unique ray that has a propagation sequence reaching the receiver. This ray is linked to the unique image source representing a given sequence of intersection. In this version, the ray does not carry a portion of the radiated source energy, it carries the full energy, and, as it is collected, it gets attenuated by the distance of propagation, according to the $1/d_{\mathcal{R}}^2$ law.

In the recent implementations of stochastic ray-based algorithms such as Sonel Mapping *Kapralos et al.* [2006], Phonon Tracing *Bertram et al.* [2005], or sound particles *Stephenson* [1990], the rays are often called particles. In order to clarify the definitions of the propagation algorithms in the following chapters, we propose the following definitions:

Definition 2.1. A particle is an entity traveling in a virtual scene that has no memory of its travel. A particle carries a portion of the energy of the sound source that is proportional to the number of particles emitted by the sound source.

Definition 2.2. A ray is an entity traveling in a virtual scene with a record of its travel, *i.e.*, the path followed by the ray between the source and its current position can be traced at any time. The ray carries the total energy of the source and gets attenuated during its travel according to the $1/d_{\mathcal{R}}^2$ law, with $d_{\mathcal{R}}$ the distance traveled by the ray.

Definition 2.3. A path represents the full trajectory of a ray or a particle from the point it is emitted to the point it is observed. A path keeps track of every position and direction (for emission, for each intersection and for the observation). It also keeps the characteristics of the walls intersected. A path can be observed at every instant of its travel. If it is observed while it reaches the receiver, the path represents the full trajectory between the source and the receiver.

In the rest of the document, all presented algorithms will use the above definitions. The original names of the algorithms will be kept unchanged. — for instance, the Monte Carlo Ray Tracing (MCRT) algorithm does not refer to the propagation of rays, but to the propagation of particles, according to the previous definitions.

The algorithms of beam tracing, cone tracing, and frustum tracing belongs to the family of ray algorithms. The difference is that the ray is no-longer a punctual entity. It defines a volume whose shape is given by the name of the algorithm.

[1] At least in the version of *Kulowski* [1985].

2.1.2 Sound emitter

From a computational point of view, the sound source can be seen as a punctual point radiating small particles[2]. These particles are infinitesimal elements traveling with the wavefront in the direction of propagation. It can be seen as a small element exploding into N_p particles that are spread omni-directionally and that continue their travel in the virtual scene. Some examples of such a representation are presented by *Deines et al.* [2006].

In all ray-based and particle-based algorithms, the source emits elements in the virtual scene. There are two ways to distribute the rays or the particles. The first is to make a deterministic discretization of the sphere. This can be for instance an exosphere, *i.e.*, a kind of equally discretized sphere, where the corners of all patches represent the direction of emission of rays or particles.

The second method is to sample the sphere using a random number generator. The random numbers define directions distributed uniformly on the surface of the sphere — generally, we use a unit sphere, as it is useful to manage unit vectors for propagation. The uniformly distributed samples $\mathbf{x}$ of Cartesian coordinates (x, y, z) on the unit sphere can be computed with two uniformly distributed parameters in the interval $[0..1]$, ξ_1 and ξ_2:

$$\begin{cases} x &= \sqrt{1-z^2}\,\cos\theta \\ y &= \sqrt{1-z^2}\,\sin\theta \quad , \qquad \theta = 2\pi\xi_2 \\ z &= 1-2\xi_1 \end{cases} \tag{2.1}$$

Tests were performed using a pseudo random number generator presented by *Wong et al.* [1997]. But no real improvements were observed in term of the convergence speed of the algorithms[3] compared to uniform distributions.

2.1.3 Sound ray/particle propagation and reflections

Once a ray or a particle was emitted, it travels in the virtual scene until it is absorbed, or until it leaves the scene (for the cases where the scene is open). Every time a particle or a ray encounters a wall, it is reflected, absorbed, or stored in a collect structure associated with the diffusion model.

In the previous sections, one parameter was omitted in the definition of the acoustic problem: the frequency. The signals used in general for auralization are broadband sounds [4] that are obtained by anechoic recording of voices, instruments or any other source present in the environment. The sound is attenuated during its propagation in the air, this attenuation depends on the frequency. During its travel in the virtual scene, the sound spectrum is also modified by the objects it interacts with. Two kinds of interactions can be observed: (i) diffraction, that is not presented in this study — The reader can refer to *Keller* [1961] for basics of the geometrical theory of diffraction, *Peter et al.* [1999]; *Calamia*

[2] More complex sound sources (*e.g.* vibrating surfaces) can be approximated by multiple point sources.
[3] This study is part of *Galletti* [2010] report.
[4] Mono frequency sounds are generally only used during the tests of the algorithms.

[2009] for Biot-Tolstoy-Medvin method, *Tsingos* [1998]; *Tsingos et al.* [2000] for specific information on the diffraction for virtual acoustic, and *Kapralos* [2006] for diffraction using Sonel mapping; (ii) the reflection of the sound on the walls of the virtual scene. As the sound reflects on a surface, it gets attenuated. That means that part of its energy is absorbed by the wall. This absorption is in general frequency dependent.

Bidirectional Reflectance Distribution Function (BRDF) f_r

As a sound wave reaches a patch of the virtual scene, it is reflected. A general method to characterize the reflection of a wave on a surface is the Bidirectional Reflectance Distribution Function (BRDF) that was quickly introduced in Section 1.1.5. Bi-directional Subsurface Scattering Reflectance Distribution Function (BSSRDF) is a more general way to model reflections, it can model more complex kinds of interactions, like waves at grazing incidence, but its study is beyond the scope of this document.

BRDF is a reflectance function, it expresses how much energy flow arriving at point $\mathbf{x}$ will leave $\mathbf{x}$. Formally, BRDF is defined as the ratio of the outgoing differential radiance for a given direction over the incoming differential irradiance (for a given direction, over an incoming differential solid angle):

$$f_r(\mathbf{x}, \mathbf{\Psi} \rightarrow \mathbf{\Theta}) = \frac{\mathrm{d}L(\mathbf{x} \rightarrow \mathbf{\Theta})}{\mathrm{d}\mathbf{I}(\mathbf{x} \leftarrow \mathbf{\Psi})} = \frac{\mathrm{d}L(\mathbf{x} \rightarrow \mathbf{\Theta})}{L(\mathbf{x} \leftarrow \mathbf{\Psi}) \cos\psi \mathrm{d}\mathbf{\Psi}} \tag{2.2}$$

where

- $\mathbf{\Theta}$ is the outgoing direction,
- $\mathbf{\Psi}$ is the incoming direction,
- ψ is the angle between the incoming direction $\mathbf{\Psi}$ and the normal at point $\mathbf{x}$.

The notations used in Equation 2.2 refer to Figure 1.2.

BRDF has the following properties:

- it can take any positive value,
- it is in general frequency dependent,
- it is a six dimension function: two for the position $\mathbf{x}$ (over the surface of the object), and two angles in polar coordinates (the incoming $\mathbf{\Psi}$ and outgoing $\mathbf{\Theta}$ directions),
- it is reciprocal $f_r(\mathbf{x}, \mathbf{\Psi} \rightarrow \mathbf{\Theta}) = f_r(\mathbf{x}, \mathbf{\Theta} \rightarrow \mathbf{\Psi})$,
- and it conserves energy; according to the first law of thermodynamic, energy can not be created, but only transformed. Our reflecting material can not add more energy into the scene, it can only decrease it. Expressing this rule in our physical quantities, we get that the outgoing power at point $\mathbf{x}$ must be smaller than the incoming power.

In this document, we limit ourselves to specular and diffuse BRDFs, discussion of other types of BRDF can be found in computer graphics books like *Pharr and Humphreys* [2004].

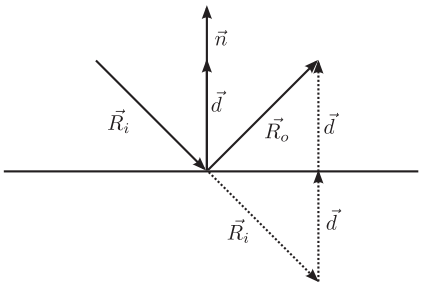


Figure 2.1: Specular reflection construction and notation.

Specular reflections

Specular reflections, also called mirror reflections, are one of the most important effects in room acoustics. They occur when a sound wave interferes with a hard surface — for instance a concrete wall. This kind of BRDF has been studied a long time before the invention of the computer — at least in optics. In 1634, Descartes gave the rules describing the reflection of light on a reflector:

- A single ray of incoming light is reflected as a single ray of outgoing light,
- the reflected ray is inside the plane defined by the incoming ray and the normal to the surface,
- the reflected ray makes with the normal an angle equal to the angle of the incoming ray.

The geometric expression of these observations is:

$$\begin{cases} \mathbf{\Psi} = \mathbf{\Theta} + 2\mathbf{d} \\ \mathbf{d} = \frac{(-\mathbf{\Theta}\cdot\mathbf{n})\mathbf{n}}{\mathbf{n}^2} \end{cases} \tag{2.3}$$

where $\mathbf{\Psi}$ is the outgoing direction of a ray, R, after reflection, and $\mathbf{d}$ is the projection of vector $\mathbf{\Theta}$, the incident direction of the ray on the normal, $\mathbf{n}$, to a reflector.

This can be expressed as a BRDF, $f_{r\mathrm{S}}$, using a Dirac delta function[5]:

$$f_{r\mathrm{S}} = \frac{\vartheta_{\mathrm{S}}}{\cos\theta_i}\delta(\cos\theta_i - \cos\theta_{\mathcal{E}})\delta(\phi_i - \phi_{\mathcal{E}} \pm \pi) \tag{2.4}$$

where ϑ_{S} is the specular reflectance of the material. The reflectance, also called sound power reflection coefficient, is defined as the ratio of reflected to incident wave power. The specular reflectance represents the portion of sound reflected in a specular direction. (θ_i, ϕ_i) and $(\theta_{\mathcal{E}}, \phi_{\mathcal{E}})$ are respectively the incident and outgoing angle of the ray in polar coordinates.

It is important to note that Equations 2.3 and 2.4 are valid for rays and particles.

[5]The definition of the Dirac delta function is given in Appendix A.3.

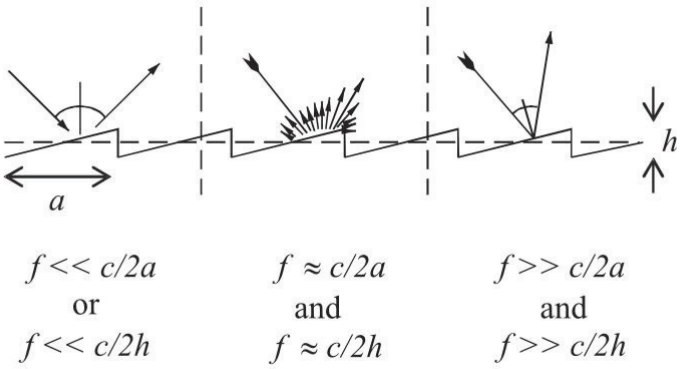


Figure 2.2: Reflection direction according to the ratio wavelength over element size — after *Cox et al.* [2006].

Lambertian (or diffuse) reflections

The diffuse BRDF is given by the formula:

$$f_{r\mathtt{D}} = \frac{\vartheta_{\mathtt{D}}}{\pi} \tag{2.5}$$

where $\vartheta_{\mathtt{D}}$ is the diffuse reflectance of the material. The diffuse reflectance represents the portion of sound reflected in diffuse direction. It is important to note that it has lost all directional dependencies: the diffuse reflectance does not depend on the incoming or outgoing direction of the ray or particle. The difficulty with diffuse reflections in ray-tracing algorithms, is that every time a ray is diffused, it should be re-emitted in an infinity of directions. This problem was solved with algorithms like radiosity, Monte Carlo Ray Tracing (MCRT), or sonel mapping (see Section 2.1.6).

Scene materials - The specular/diffuse model

One of the most used model in room acoustic simulation is the combination of specular and diffuse models. It has the major advantage to be easily compatible with particle algorithms, and requires less computation than complete BRDF algorithms. Some methods like *Tsingos et al.* [2007] were developed in order to estimate the BRDF of materials and perform propagation. But, usually, the BRDFs of the materials used in the simulations are unknown[6].

Scattering occurs when the corrugations of the surface are of the order of $\lambda/2$, with λ the wavelength of the sound propagated (see Figure 2.2). That means that the type of reflection is frequency dependent. In the specular/diffuse model, the BRDFs are approximated by a combination of specular and

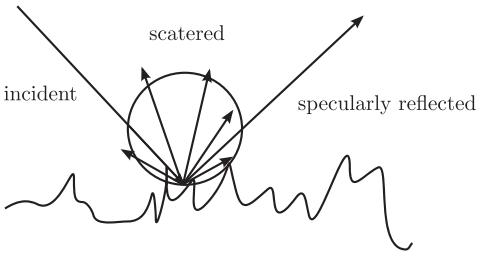


Figure 2.3: Combining Lambertian and specular BRDF.

Lambertian BRDFs, as shown in figure 2.3.

The energy reflected specularly ($\Phi_{\mathtt{S}}$), diffusely ($\Phi_{\mathtt{D}}$) and the total reflected energy (Φ_{tot}) are

$$\Phi_{\mathtt{S}} = \Phi_i(1-\alpha)(1-s) = \vartheta_{\mathtt{S}}\Phi_i \tag{2.6}$$

$$\Phi_{\mathtt{D}} = \Phi_i(1-\alpha)s = \vartheta_{\mathtt{D}}\Phi_i \tag{2.7}$$

$$\Phi_{tot} = \Phi_{\mathtt{S}} + \Phi_{\mathtt{D}} = \Phi_i(1-\alpha) \tag{2.8}$$

where Φ_i is the incident energy, α is the Sabine absorption coefficient [see *Kuttruff*, 1973], s is the scattering coefficient — it defines the portion of the reflected sound that is diffused. $\vartheta_{\mathtt{S}}$ is the portion of energy reflected in specular direction, and $\vartheta_{\mathtt{D}}$ is the portion of energy diffused. Databases of frequency dependent Sabine (α) and diffuse coefficient (s) can be found in *e.g.* *Bork* [2005a] and *Vorländer* [2008]. Some recent investigations on the determination of α and s were presented by *Hanyu* [2010].

2.1.4 Sound receiver

Rays or particles collection is the link between geometric processing and auralization. This section presents various structures used in the literature to collect rays or particles. It describes what kind of elements are collected by the receiver and how they are arranged. Various receiver characteristics will lead to various auralization methods, or to various improvements in the propagation algorithms. The analysis of the pros and cons of six structures is presented in this section. An alternative way of collecting rays based on perceptive information is presented in Chapter 4. But, first of all, we will present the differences related to the collection of rays or particles.

Ray versus particle collect algorithms

The processing of a ray or a particle is very different after the collect step. The propagation algorithms for rays or particles may share some characteristics. But, once the propagated entities are detected, the algorithms become really different. We present here the basic information about the collect of these two entities. The steps following the collect are explained in Chapter 3.

[6] In some special cases, analytic formulations can also be found.

Ray collect As we have seen earlier, the rays traveling in the virtual scene keep track of their history, *i.e.*, the set of all reflections encountered during their travel. When they reach the receiver, they have traveled a certain distance, $d_{\mathcal{R}}$, from the source, and have an associated energy, $\Phi_{\mathcal{R}}$. The direction of arrival on the receiver, (R_θ, R_ϕ), is also kept. When a ray reaches the receiver, it is possible to add its contribution directly to the RIR as the ray carries the full energy emitted by the source. Figure 2.7(d) presents the collect of pure specular rays in the scene of Appendix B.1.1 up to order five. A ray can be located in time very precisely, even if the receiver is not punctual, as it is possible to detect the corresponding image source thanks to the ray history.

Particle collect The method used for collecting particles is very different from that for rays. As a particle reaches a receiver, the energy it carries is added to an integrated echogram (a part of the receiver's energy proportional to the number of particles emitted). If the integration steps of the echogram are too small, the particles will be spread in many bins of the echogram, and the specular contributions will not be perfectly located in time. If the integration steps are too high, many paths coming from various reflections may be summed and the echos or interferences caused by specular paths may be lost. Figure 2.10 shows three integrated echograms with different integration steps. Chapter 4 discusses the choice of the integration steps from a perceptive point of view.

* * *

The collect structures presented in the current section are suited for particle or ray algorithms. They present various ways to improve collect of the rays or particles propagation.

Fixed size structures (for particles)

The first structures that were implemented can be classified as fixed size structures. *Krokstad et al.* [1968]; *Kulowski* [1985]; *Vorländer* [1989] used such structures to implement the first versions of particle tracing algorithms in acoustics.

The main problem in particle tracing algorithms is that a particle is infinitely small, and cannot be detected by an infinitely small sensor, *i.e.*, a collecting structure. Different types of sensors can be implemented, depending on the applications. *Kulowski* [1985] presents in details the two simplest types of structures to collect paths. The first one is a plane, that can be seen just like all other triangulated parts of the virtual scene. Each time a particle reaches this structure, it is collected in the echogram to form the impulse response of the scene. This structure has the obvious disadvantage of being very directional: particles parallel to this plane will be lost. The second collect structure is a sphere centered on the receiver point. This structure is very often used in ray or particle tracing algorithms. It has the main advantage of being an omni-directional sensor, and collect can be simply implemented by computing the particle/sphere intersection. This is written as:

$$\begin{cases} (x_{\mathcal{I}} - x_{\mathcal{R}})^2 + (y_{\mathcal{I}} - y_{\mathcal{R}})^2 + (z_{\mathcal{I}} - z_{\mathcal{R}})^2 = r_{\mathcal{R}}^2 \\ \mathbf{x}_{\mathcal{I}} = \mathbf{x}_{\mathbf{P}} + t.\mathbf{d}_{\mathbf{P}} \end{cases} \tag{2.9}$$

where $x_{\mathcal{R}}$, $y_{\mathcal{R}}$ and $z_{\mathcal{R}}$ are the coordinates of the center of the receiver sphere, $r_{\mathcal{R}}$ the sphere radius, $\mathbf{x}_{\mathcal{I}}$, the intersection point of coordinates $(x_{\mathcal{I}}, y_{\mathcal{I}}, z_{\mathcal{I}})$, $\mathbf{x_P}$ and $\mathbf{d_P}$, the incoming particle origin and direction respectively[7]. The number of rays used in a simulation for a sphere of radius $r_{\mathcal{R}}$ [after *Vorländer*, 1989]:

$$N_p \geq \frac{4(ct_{max})^2}{r_{\mathcal{R}}^2} \tag{2.10}$$

with t_{max}, the duration of the echogram to generate and c, the speed of sound. This equation shows that to generate an echogram of one and a half seconds with a sphere of radius one meter, around 10^6 particles have to be propagated. With this structure, the collected particles increase the energy of the echogram bin associated with their arrival time.

The problem with the previous structure is that it only stores the energy of the incoming particle reaching the sensor: the incoming directions are lost. The resulting echogram will therefore allow monophonic rendering only.

Fixed size structure with history mechanism (for rays)

This structure is an extension of the previous one to ray propagation algorithms. It is used for instance in Deterministic Ray Tracing (DRT) (see Section 2.1.5). In this algorithm, the propagated ray stores a history of all surfaces encountered during its travel in the scene. The collect structure has the same geometry as the one described in the previous section. But, every time a ray is collected, it is compared with all previously collected rays. If one of the previously collected rays has the same history, only one of the rays is kept[8].

With this structure, the energy Φ_R carried by every ray R is

$$\Phi_R = \frac{1}{d_R^2} \prod_{i=1}^{N_{refl}} (1 - \alpha_i) \tag{2.11}$$

where d_R is the distance traveled by ray R, N_{refl}, the number of reflections that occurred before the ray was captured, and α_i the Sabine absorption coefficient for the i^{th} intersected wall.

There are two drawbacks to this method: (i) the comparison between the current ray and the previously collected rays is of order $\mathcal{O}(N_{rays}^2)$; (ii) the only way to manage curved surfaces is to discretize them, *i.e.*, curved surfaces lose their continuity. They are triangulated, and a source image is associated to every triangle.

Fixed size structures with spatial information (for particles)

Fixed size structures are still very often used. *Lentz et al.* [2007] use a sphere sensor and *Röber et al.* [2007] use a cube. Recently, the concept of the fixed size structure was extended to capture the incoming

[7] $\mathbf{x}_{\mathcal{I}} = \mathbf{x_R} + t.\mathbf{d_R}$ for a ray.

[8] This is the ray with the smaller arrival time, as specular ray reflections minimize the distance between the source and the receiver after N_{refl} reflections. The other rays are discarded.

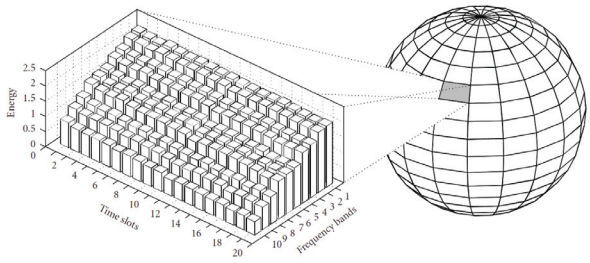


Figure 2.4: An echogram of a collect sphere with spatial information — after *Lentz et al.* [2007].

directions of the rays. *Röber et al.* [2007] introduced a structure called cube-map which is a discretized cube where each cell stores the incoming rays energy and delay per frequency band. The stored data are used to obtain the arrival direction in order to produce a binaural synthesis. One of the problems with this structure is that it is not omnidirectional, some directions are privileged during the collect.

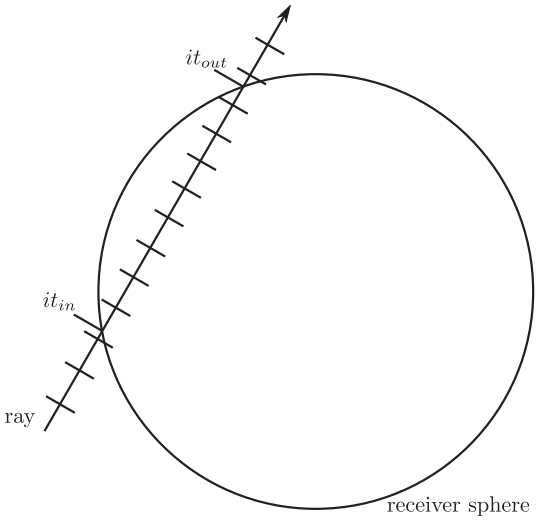
Another approach of spherical collection is proposed by *Lentz et al.* [2007]. The principle is the same as the cube collect structure, except that the sensor here is a discretized sphere. Figure 2.4 shows that the sphere gathers an echogram — energy and delay per frequency bands — for each cell of the sphere. A small problem with this structure, is that the discretization is not regular. This has two disadvantages. First, the poles of the sphere will provide a better location of the sound because the patches are smaller in these regions — this is not adapted to human perception. Second, some cells on the poles will have a very small probability to collect rays, this will lead to empty echograms, that can consume unused memory.

The above structure, with a regular discretization, like an exosphere could be a good choice for fixed size structure algorithms. Other strategies can be used for the discretization, like a discretization based on perceptive parameters. This is the approach proposed in Chapter 4.

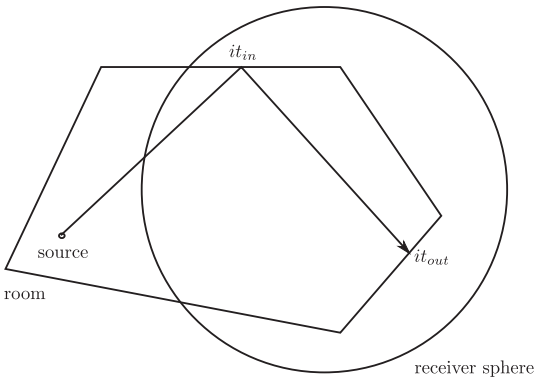
Adaptive structures (for particles)

Recently, *Lesoinne and Embrechts* [2008] and *Lesoinne* [2006] presented a method based on an adaptive spherical receptor in order to accelerate particle tracing methods. In this method, the size of the receiver increases according to the propagation distance of the rays in the virtual scene. In order to keep a low statistical error, either the number of particles or the size of the receiver increases when t_{max} increases (*cf.* Equation 2.10). This method can be seen as the generation of several echograms, each of them associated with increasing t_{max}. In their method *Lesoinne and Embrechts* [2008] adapt the number of rays traveling in the scene according to the size of the receiver. They also describe a growing receiver that can be larger than the room dimensions when dealing with late reflections. The growing factors and temporal steps are arbitrarily determined by the authors. The collect algorithm is modified as follows:

A temporal spreading of the particles contributions is introduced: If the collect structure is longer than the temporal resolution of the echogram, the particle energy is spread over time intervals $[t_{in}, t_{out}]$,



(a) Ray temporal spread when crossing a large receiver. The ray energy is spread over time intervals between t_{in} and t_{out}.



(b) Ray intersection times when receiver is larger than room dimensions.

Figure 2.5: Two rays intersecting a large collect structure.

with t_{in}, the incoming and t_{out}, the outgoing intersection times with the sphere.

When the collect sphere becomes larger than the room, intersection times are taken at the wall position as shown in Figure 2.5. The same temporal spreading mechanism is applied between these two points.

To keep a constant particle/receiver intersection probability, the number of rays is decreased each time the sphere grows up. When the sphere dimensions change, particles are terminated, *i.e.*, they are no longer propagated in the scene. When the radius of the sphere grows from r_1 to r_2, the new number of particles in the scene, N_{p2}, is:

$$N_{p2} = N_{p1} \left(\frac{r_1}{r_2} \right)^2 \tag{2.12}$$

To keep a constant global energy of the rays at any given time, the energy of the N_{p2} new rays must be multiplied by N_{p1}/N_{p2}. If the receiver size is larger than room dimensions, its equivalent radius is determined using an offline preprocessing step as explained in *Lesoinne and Embrechts* [2008].

More information on the implementation of such a mechanism can be found in *Hermant* [2010].

Other types of collect methods

In this section, we have presented the collect structures commonly used for ray or particle collect algorithms. Depending on the propagation algorithms, other collect structures can be implemented. For algorithms that propagate beams, pyramids or cones, the receiver is a punctual entity. The detection is then made by testing if the point corresponding to the receiver's position is inside or outside the volume of the propagated element (cone, beam, frustum, ...).

Another way of collecting the information propagated from the sound emitter is to perform a second ray or particle shooting from the receiver following the shooting from the emitter. These algorithms are called bidirectional algorithms, and the receiver in this case is also punctual. The detection of the paths is made every time a ray or a particle starting from the receiver intersects a wall. At every intersection, the collect of the information propagated from the source (and stored on the walls) is performed. Sonel mapping (see Section 2.1.6) is an example of such an algorithm.

2.1.5 Pure specular contributions to the sound field

In room acoustics, the most significant paths from a perceptive point of view are the first specular reflections. In this section, we also include the direct sound in the set of specular paths as it can be seen as a specular path without reflection, or a zero order specular path. The first orders of specular paths carry information about the direction of the sound, as well as perceptive information about the size of the scene. Specular reflections of higher orders also carry important information on the characteristics of the room. For instance, in a perfect shoe box room, a coloration of the sound or a flutter echo can be heard when rendering specular paths of at least three orders. In this section, we present two algorithms specialized in the detection of specular paths in the virtual scene. The image source algorithm and the Deterministic Ray Tracing (DRT) algorithm.

Image source algorithm

Algorithm 1: Image source recursive algorithm.

Data: Image Source $\mathcal{E}_i$
Receiver $\mathcal{R}$
Wall w_{prec}
Reflection order o
begin `ImageSourceRec` $\{\mathcal{E}_i, \mathcal{R}, w_{prec}, o\}$
  if $o = N_{refl}$ **then**
    return
  else
    foreach $w \in walls, w \neq w_{prec}$ **do**
      $\mathcal{E}_{i+1} =$ `MirrorImage`$(\mathcal{E}_i, w)$
      `BackwardRayTracing` $(\mathcal{R},\mathcal{E}_{i+1})$
      `ImageSourceRec` $(\mathcal{E}_{i+1},\mathcal{R},w,o+1)$

The image source algorithm presented by *Allen and Berkley* [1976] is the simplest way to find exhaustively all specular paths between a source and a receiver in a virtual scene. The first order of image sources is obtained by mirroring the source with all the walls of the scene. The second order of image sources is generated by mirroring all the first order image sources with all the walls of the scene except the one that generated second order image sources. The same scheme is used for the following

N_{walls} \ N_{refl}	1	10	20	50	100
6	7	$1.2 \cdot 10^{7}$	$1.2 \cdot 10^{14}$	$1.1 \cdot 10^{35}$	$9.9 \cdot 10^{69}$
50	51	$8.1 \cdot 10^{16}$	$6.5 \cdot 10^{33}$	$3.3 \cdot 10^{84}$	$1.0 \cdot 10^{169}$
100	101	$9.1 \cdot 10^{19}$	$8.3 \cdot 10^{39}$	$6.1 \cdot 10^{99}$	$3.7 \cdot 10^{199}$
1000	1001	$9.9 \cdot 10^{29}$	$9.8 \cdot 10^{59}$	$9.5 \cdot 10^{149}$	$9.1 \cdot 10^{299}$

Table 2.1: Number of image sources depending on the number of walls and the order of reflections.

orders of reflections. The problem with this method is its exponential complexity. Table 2.1 gives the number of image sources calculated for various orders of reflections and number of walls in a virtual scene. The exact number of image sources, N_{IS}, depending on the number of non coplanar walls in the scene, N_{walls}, and the order of reflections, N_{refl}, is

$$N_{IS} = 1 + N_{walls} + \sum_{i=2}^{N_{refl}} (N_{walls} - 1)^{i} \tag{2.13}$$

One problem remains with the algorithm of image sources; not all of the image sources constructed are visible from the receiver. Every time the processing is applied to complex scenes — at least more complex than shoe box rooms — some of the image sources are hidden. They can be hidden by an occluder, or by a complex element of the geometry. *Borish* [1984] proposed a method for extending the calculation of image sources to arbitrary polyhedral environments. In this method, for all newly created image sources, a test is performed to check if it is in a visibility zone from the receiver or not. One of the most used methods to determine whether the image source is valid or not is to perform a backward ray-tracing from the receiver, and check if the ray reaches the source. The algorithm of ray tracing is similar to the one described in the next section. In order to check occlusion, the ray is created at the position of the receiver, and is oriented in the direction of the image source to check. The ray is then propagated in the scene upon the same order than the image source. If the ray reaches the receiver after the last specular reflection in the scene, then the image source is valid, and can be kept for the calculation of the IR. Figure 2.6 illustrates an image source creation for a simple 2D geometry, and the back-traced ray to check if one of the sources is valid. In this figure, the studied room is represented by solid lines, the other rooms in dash lines contain the image sources, they are called the virtual rooms by *Borish* [1984].

The main advantage of the image source algorithm is that it gives a totally exhaustive list of the specular paths up to the considered reflection order N_{refl}. We have implemented this algorithm in order to build the tree of reflections, a tree structure containing all the image sources contained in a virtual scene (see Section 2.1.7).

Algorithm 1 shows the recursive procedure to find the image sources in an arbitrary room. This

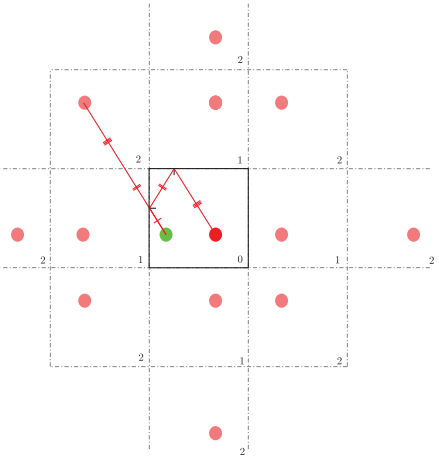


Figure 2.6: Image sources of order zero, one and two, and a back-traced ray for one of them.

algorithm includes the back-tracing method to check the validity of the image source, *i.e.*, no occluder exists between the image and the receiver, and they can be connected with a specular ray. The position of the image source $\mathbf{x}_{\mathcal{E},\mathbf{i+1}}$[9] after the mirror reflection is (after *Borish* [1984])

$$\mathbf{x}_{\mathcal{E},\mathbf{i+1}} = \mathbf{x}_{\mathcal{E},\mathbf{i}} + 2d\mathbf{n}, \qquad d = -\mathbf{n} \cdot (a - \mathbf{x}_{\mathcal{E},\mathbf{i}}) \tag{2.14}$$

where a is any point on the wall, d, the projection of the vector $a - \mathbf{x}_{\mathcal{E},\mathbf{i}}$ on the normal to the wall.

Deterministic ray tracing

One of the most important algorithms in geometrical acoustics is the ray tracing algorithm. It is important in the sense that it defines the basic concept for most of the propagation algorithms presented in this research. The main algorithms derived from the ray-tracing algorithm in acoustics are:

- the stochastic ray tracing algorithm — *Dalenbäck* [1996];
- the beam tracing algorithm — *Funkhouser et al.* [2004];
- the cone tracing algorithm — *van Maercke and Martin* [1993];
- the sonel mapping algorithm — *Kapralos* [2006];
- the phonon tracing algorithm — *Bertram et al.* [2005];
- the acoustic radiance transfer — *Siltanen et al.* [2007].

There are still possible improvements in geometrical acoustics, following progress in the field of computer graphics. In particular ray-tracing algorithms were declined in various versions, such as:

[9] $\mathbf{x}_{\mathcal{E},\mathbf{i+1}}$ represents an image of the source $\mathcal{E}$. As the source corresponds to an image of order zero, it could also be noted $\mathbf{x}_{\mathcal{E},\mathbf{0}}$.

- the path tracing algorithm — *Kajiya* [1986];
- the bidirectional path tracing algorithm — *Lafortune and Willems* [1993];
- the photon mapping algorithm — *Jensen* [2001]; [10]
- the metropolis ray tracing algorithm — *Veach and Guibas*.

Ray tracing is a geometric method that connects a source and a receiver. Sources and receivers are located in an environment where sound propagates. Ray tracing algorithms are based on Fermat principle, also called the principle of least time, that is the path taken between two points by a ray is the path that can be traveled in the least time. As we have seen before, the travel time is directly proportional to the distance traveled by a sound wave, so a ray is defined as the minimal distance between two points after N_{refl} reflections. This principle yields the very important property that the traveling direction of a ray has no influence on the simulation, *i.e.*, a ray starting from the source and reaching the receiver will have the same characteristics if the source and the receiver were permuted.

The first simulations in the field of room acoustics were made using a stochastic method to sample the distribution of the rays in a virtual scene [*Allred and Newhouse*, 1958a,b]. This is the first article about Monte Carlo Ray Tracing (MCRT) algorithm (see Section 2.1.6 for more information on MCRT). This method was used to estimate acoustical properties such as reverberation time or mean free path in parallelepipedic rooms.

In the DRT algorithm, the rays are thrown from the source, then reflected on the walls of the scene as purely specular contributions and finally collected by the receiver. The receiver is in general a sphere, in order to have an omnidirectional collecting element (see Section 2.1.4). According to Equation 2.10, the number of rays, N_{rays}, [11] to throw depends on the receiver radius, $r_{\mathcal{R}}$, and the maximum time of the echogram, t_{max}, according to the relation

$$N_{rays} \geq \frac{4(ct_{max})^2}{r_{\mathcal{R}}^2} \tag{2.15}$$

Algorithm 2 presents a simple version of the DRT algorithm, where the termination criterion, *i.e.*, the time a ray disappears from the scene, is based on the energy of the ray. In the DRT algorithm, the termination criterion could be either the travel time of the ray, or the remaining energy of the ray. In the first part of the algorithm, the ray position, $\mathbf{x_R}$, direction, $\mathbf{d_R}$, distance, d_R, and energy, Φ_R, are initialized. Then the ray is propagated until its energy falls below a termination threshold, $\varepsilon_{\mathcal{E}}$. When the ray travels in the scene, each intersection is characterized by an intersection object $\mathcal{I}$. Depending on the kind of intersection, the following behaviors occur.

(i) If the ray crosses a receiver, and no other ray collected by the receiver has the same history, then the ray is stored by the receiver.

[10] That is the basis for sonel and phonon algorithms in acoustics.
[11] Here, N_{rays} is equivalent to the number of particles N_p.

Algorithm 2: Deterministic Ray Tracing (DRT) algorithm.

```
foreach Ray R do
    x_R ← x_E
    d_R ← UniformSampleSphere()
    d_R ← 0
    Φ_R ← Φ_E
    while Φ_R / R_R^2 ≥ ε_E do
        I = GetNearestIntersection()
        if I_type is Receiver then
            if CheckUnique (R) then
                Collect (R)
        else if I_type is ∞ then
            continue
        else if I_type is Wall then
            x_R ← I_pos
            d_R ← SpecularReflection(R, I)
            Φ_R ← Φ_R · α_I
            d_R ← d_R + d_I
            R_history ← R_history | I
```

(ii) If the scene is opened, the ray may miss a surface and go to infinity, here, the ray stops its travel, and disappears from the scene.

(iii) If the ray intersects a wall, then its new position is the intersection point, $\mathbf{x}_{\mathcal{I}}$, its new direction is a specular reflection (see 2.1.3), its energy gets attenuated by the Sabine coefficient of the surface, $\alpha_{\mathcal{I}}$, and its travel distance is incremented by the distance between the last and current intersection. Finally, its history, $R_{history}$, keeps a history of the reflection. In Algorithm 2, the symbol | represents the concatenation operator.

We present in Section 2.5.1 an alternative implementation of DRT algorithm that is suited to the problem of real time auralization.

2.1.6 Diffuse reflections

We have seen in Section 2.1.3 that the specular model is not sufficient to describe the propagation of a sound wave in a complex environment. In this section, we present the radiosity algorithm that can be used to estimate the diffuse sound field in a room. It is impossible with classical radiosity algorithms to obtain the specular reflections between source and receiver. *Tsingos* [1998] proposed a progressive radiosity algorithm that is suited to the problem of specular paths in radiosity algorithms. The presentation of this algorithm is beyond the framework of this study. Instead, we decided to implement a mixed version of the classical radiosity algorithm and image source method (see Section 2.3). We then present methods based on Monte Carlo Ray Tracing (MCRT) to calculate both the specular and the diffuse part of the propagation using ray based algorithms.

Radiosity

The radiosity technique was first used in the 50's to calculate thermal propagation. It has then been adapted in various fields of research, such as electromagnetism, computer graphics, and acoustics. The principle of radiosity is based on the propagation of radiance in the virtual scene. The formulation of the total radiance L emitted by a point $\mathbf{x}$ of an element of the scene in direction $\mathbf{\Theta}$ (from Equation 1.12) is

$$L(\mathbf{x} \to \mathbf{\Theta}) = L_e(\mathbf{x} \to \mathbf{\Theta}) + \int_{\mathcal{G}} f_r(\mathbf{x}, \mathbf{\Psi} \to \mathbf{\Theta}) L(\mathbf{x}' \to -\mathbf{\Psi}) V(\mathbf{x}, \mathbf{x}') G(\mathbf{x}, \mathbf{x}') \mathrm{d}\mathbf{x}' \tag{2.16}$$

If we consider here the particular case of Lambertian surfaces with a time independent BRDF (see Equation 2.5), we then have

$$L(\mathbf{x}) = L_e(\mathbf{x}) + \frac{\vartheta_{\mathtt{D}}}{\pi} \int_{\mathcal{G}} L(\mathbf{x}' \to -\mathbf{\Psi}) V(\mathbf{x}, \mathbf{x}') G(\mathbf{x}, \mathbf{x}') \mathrm{d}\mathbf{x}' \tag{2.17}$$

where

- $L(\mathbf{x})$ is the total radiance leaving point $\mathbf{x}$ at the instant t;
- $L_e(\mathbf{x})$ is the radiance emitted by the patch at point $\mathbf{x}$ at time instant t;
- $L(\mathbf{x}' \to -\mathbf{\Psi})$ is the incident radiance at point $\mathbf{x}$ and instant t from direction $-\mathbf{\Psi}$;
- $\vartheta_{\mathtt{D}}$ is the diffuse reflectance of the patch.
- $V(\mathbf{x}, \mathbf{x}')$ is the visibility function, it is a boolean function that as a value one if the two points $\mathbf{x}$ and $\mathbf{x}'$ see each others, zero otherwise.
- $G(\mathbf{x}, \mathbf{x}')$ is a geometric term (see Section 1.1.5).

In the special case of radiosity, the total intensity $I_i(t)$ leaving a patch i is proportional to the direction independent radiance $L(t)$:

$$I(t) = \pi L(t) \tag{2.18}$$

The radiosity technique can be formulated by discretizing the integral over all surfaces of the scene by a sum of all radiative exchanges between patches. For patch i, this is expressed as:

$$I_i(t) = I_{e,i}(t) + \vartheta_{\mathtt{D}} \sum_{j=1}^{N_{patch}} F_{i \to j} I_j(t - t_{ij}) \tag{2.19}$$

where

- I_i, the intensity leaving patch i;
- $I_{e,i}$, the intensity emitted by patch i;
- $\vartheta_{\mathtt{D}}$, the diffuse reflectance of the material;

- I_j, the intensity emitted by patch j;
- t_{ij}, the propagation time between the center of the patches i and j.

Symbol $F_{i\to j}$ represents the form factor that measures the proportion of energy leaving patch i that reaches patch j:

$$F_{i\to j} = \frac{1}{A_i}\int_{A_i}\int_{A_j} \frac{\cos\Psi_1 \cos\Psi_2}{\pi\ |\mathbf{x}-\mathbf{x}'|^2} V(\mathbf{x},\mathbf{x}')\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}' \tag{2.20}$$

where

- Ψ_1 is the angle between vector $\mathbf{x}\mathbf{x}'$ and the normal at point $\mathbf{x}$;
- Ψ_2 is the angle between vector $\mathbf{x}'\mathbf{x}$ and the normal at point $\mathbf{x}'$;
- A_i is the area of patch i;
- V is the visibility function.

The notations refer to Figure 1.3. For the specific case of exchanges between a punctual point in the scene and a surface, the form factor becomes

$$F_j = \frac{1}{A_j}\int_{A_j} \frac{\cos\Psi_1}{4\pi||\mathbf{x}-\mathbf{x}'||^2} V(\mathbf{x},\mathbf{x}')\mathrm{d}\mathbf{x}' \tag{2.21}$$

It is important to underline the fact that in acoustics, the form factors are time dependent, as the radiative exchanges between patches cannot be considered as instantaneous.

The calculation of the form factors will be detailed in Section 2.3, where a step by step description of the hybrid image source/radiosity algorithm is presented.

Monte Carlo Ray Tracing (MCRT)

We have presented previously DRT algorithm (see Section 2.1.5). This section introduces ray tracing algorithms based on Monte Carlo integration. The mathematical theory about Monte Carlo integration can be found in Appendix A.7. Recalling from the definitions given in Section 2.1.1, this algorithm is considered as a particle algorithm. So, in the rest of the document, the basic propagation element of the MCRT algorithms will be particles, and not rays as suggested by the name of the algorithm.

We present here a version of MCRT that takes into account specular and diffuse reflections. Although the algorithm is stochastic, the particles can be generated either by stochastic or deterministic patterns as described in Section 1.1.3. The particles travel with a portion of the energy, Φ_P, proportional to the number of particles emitted by the source, N_p:

$$\Phi_P = \frac{\Phi_{\mathcal{E}}}{N_p} \tag{2.22}$$

where, $\Phi_{\mathcal{E}}$ is the energy of the source. Every time a particle intersects a surface of the virtual scene, it is reflected according to the properties of the material (see Section 2.1.3). Different behaviors can be

applied in order to simulate the material BRDF. Here, we restrict the BRDF to the specular/diffuse model. In case of a simple specular reflection, the resolution is simple, as the particle gets reflected in the mirror direction (see Equation 2.3), and gets attenuated by α, the reflection coefficient of the intersected wall material. In case of a simple diffuse reflection, the problem is more complex, as the sound is reflected in every direction from the surface. An accurate implementation can be made by sampling the hemisphere above the surface for each new diffuse reflection. This implies throwing N_{p2} new particles every time a diffuse surface is intersected. The problem with this approach is that the number of particles in the scene will increase exponentially, and that it will be difficult to reach high orders of reflections. Another method is to generate a random direction on the hemisphere above the surface. If the number of initially thrown particles, N_p, is big enough, the result will also converge. The direction of reflection can then be computed — uniformly sampled in local polar coordinates (θ_P, ϕ_P) — as [after *Pharr and Humphreys*, 2004]

$$\begin{cases} \theta_P &= 2\pi\xi_1 \\ \phi_P &= \arccos(\sqrt{\xi_2}) \end{cases} \tag{2.23}$$

where, ξ_1 and ξ_2, are uniformly generated random numbers, θ_P is the rotation angle of the particle around the normal and ϕ_P is the angle with the normal.

Finally, if the material has both specular and diffuse properties, the previous methods are combined. We can:

- generate one particle in specular direction, and N_{p2} particles to sample diffuse space,
- or, generate one particle in specular direction, and one random particle to model the diffusion,
- or, keep only one particle, and determine if it is specular or diffuse.

The last solution is the most interesting as it keeps the number of particles traveling in the scene constant. The type of reflection is determined randomly according to:

$$\begin{cases} \xi \in [0 \dots s] &\rightarrow \textit{diffuse reflection} \\ \xi \in]s \dots 1] &\rightarrow \textit{specular reflection} \end{cases} \tag{2.24}$$

where, s, is is the scattering coefficient defined in Section 2.1.3 and ξ follows a uniform distribution between zero and one.

Finally, to reduce the number of particles traveling in the scene, the principle of Russian roulette can be applied [*Kapralos et al.* [2005]]. Russian roulette (see Appendix A.7.1) is an unbiased statistical method that reduces the number of particles traveling in the scene as the order of reflection increases. The counterpart of this method is that it increases the variance of the solution as the number of particles

decreases. The method is also based on a random number generation to determine the type of interaction:

$$\begin{cases} \xi \in [0 \dots \vartheta_{\mathtt{D}}] & \rightarrow \quad \textit{diffuse reflection} \\ \xi \in]\vartheta_{\mathtt{D}} \dots \vartheta_{\mathtt{S}} + \vartheta_{\mathtt{D}}] & \rightarrow \quad \textit{specular reflection} \\ \xi \in]\vartheta_{\mathtt{S}} + \vartheta_{\mathtt{D}} \dots 1] & \rightarrow \quad \textit{absorption} \end{cases} \tag{2.25}$$

where, $\vartheta_{\mathtt{D}}$ and $\vartheta_{\mathtt{S}}$ are respectively the specular and diffuse reflectance (see Section 2.1.3).

When a particle is absorbed, it disappears from the virtual scene, and the algorithm processes the next particle. The main disadvantage of the Russian Roulette is the increase of the variance toward the end of the echogram as the number of particles that contribute becomes small. If the initial number of particles generated is too small, the resulting echogram will have empty bins that lead to audible artifacts. Therefore, we chose not to implement the Russian Roulette in our algorithms.

The collect phase of the particles is done by one of the structures presented in Section 1.1.4. In the case of a simple collect sphere, each particle intersecting the sphere is summed in the echogram that will be later used for auralization.

The major problem with the MCRT algorithm is that its behavior is frequency dependent, *i.e.*, the behavior of the particles when they interact with a wall depends on the frequency characteristics of the material. The method commonly used to implement a frequency dependent processing is to repeat the simulation for each frequency band. The resulting echograms are frequency dependent and each of them represent the energy exchanges for a given frequency band[12]. *Embrechts* [2000] proposed a modification of the MCRT algorithm where the particles carry the information to reconstruct the frequency dependent echograms. This algorithm uses a new coefficient called the splitting coefficient, β, that is identical for all frequency bands — so that only one particle shoot is necessary. In order to keep an unbiased simulation, every time a particle intersects a wall, a corrective coefficient, C_n, is applied to each frequency band. For the n^{th} reflection, the coefficient is given by

$$\begin{cases} C_n(f) = s/\beta_n, & \textit{for a diffuse reflection} \\ C_n(f) = (1-s)/\beta_n, & \textit{for a specular reflection} \end{cases} \tag{2.26}$$

The validity of the algorithm does not depend on the value of β_n, but the rate of convergence does. *Embrechts* [2000] showed that the best results were achieved using the following expression for the coefficient, β_n:

$$\beta_n = \frac{\max_f \left[\left(\prod_{i=0}^{n-1} C_i(f) \right) s(f) \right]}{\max_f \left[\left(\prod_{i=0}^{n-1} C_i(f) \right) s(f) \right] + \max_f \left[\left(\prod_{i=0}^{n-1} C_i(f) \right) (1 - s(f)) \right]} \tag{2.27}$$

The definition of β_n is thus reflection iterative (C_i represents the corrective coefficient for the previous reflections), it is the value that minimizes the maximum of the C_n coefficients in both specular and

[12] Usually octave bands or $1/3^{rd}$ octave bands.

diffuse case. Finally, the choice of the direction of the particle is made similarly to equation 2.24 with the splitting coefficient β:

$$\begin{cases} \xi \in [0 \dots \beta] & \rightarrow & \textit{diffuse reflection} \\ \xi \in]\beta \dots 1] & \rightarrow & \textit{specular reflection} \end{cases} \tag{2.28}$$

Sonel mapping

Algorithm 3: Sonel emission.

```
foreach Particle P do
    continue = True
    while continue do
        I = GetNearestIntersection()
        ξ = Rand[0...1]
        if ξ ∈ [0...ϑ_D] then
            AddToSonelMap (P)
            DiffuseReflection (P)
        else if ξ ∈ ]ϑ_D...ϑ_D + ϑ_S] then
            SpecularReflection (P)
        else
            continue = False
```

Algorithm 4: Sonel collect.

```
foreach Ray R do
    continue = True
    while continue do
        I = GetNearestIntersection()
        if I_type is Emitter then
            CollectSpecularRay (R)
        else
            ξ = Rand[0...1]
            if ξ ∈ [0...α_D] then
                < P > = CollectKNearestSonnels(I)
                AddSonelsToEchogram (< P >)
            else if ξ ∈ [ϑ_D...ϑ_D + ϑ_S] then
                SpecularReflection (R)
            else
                continue = False
```

Sonel mapping is an extension of the MCRT methods proposed by *Kapralos* [2006] and based on the photon mapping by *Jensen* [2001] used in computer graphics. Sonel mapping is a two pass algorithm. In a first pass, the particles are emitted by the sound source. The propagation of the particles in the virtual scene is the same as in classical MCRT algorithms with Russian roulette. During the first pass of propagation, no particle is collected by the receiver, but the particles are stored in a structure called sonel map every time they reflect diffusely from a wall. The sonel map stores the particles at the position where they reflected on the wall. Kd-tree structures are commonly used to implement sonel (or photon) maps (see *Jensen* [2001]). It has to be noted that if the photons are ordered in a kd-tree, the research of the k nearest particles in a set of N_p is made in $\mathcal{O}(k \log N_p)$. This property is used during the second pass of sonel mapping.

The second pass, referred to as the collect pass, is based on ray tracing. Rays are traced from the receiver. If they reach the sound source, this means that a specular reflection must be added to the final echogram. When a ray intersects a wall, a reflection test is performed according to Equation 2.24 to determine whether the ray is reflected specularly or diffusely. Every time a ray is specularly reflected, it continues its travel in the virtual scene, and its intersection with the sound source is tested. If the ray is diffused, the k nearest sonels are collected from the sonel map and added to the echogram; the ray terminates its travel. If the ray is absorbed, it also terminates its travel. Algorithms 3 and 4 describes the two passes of the sonel mapping algorithm.

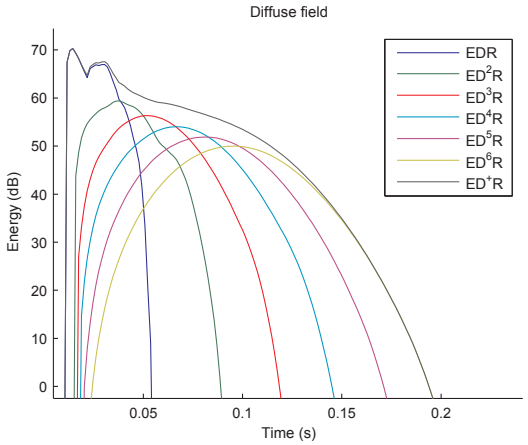
One of the major advantage of the sonel mapping algorithm is that the propagation pass can be kept as long as the sound source is immobile in the virtual scene. The particles propagated during this pass will not be modified until the source moves. In case of a moving source and receiver, the two passes have to be re-executed, and the benefits of this algorithm are lost.

2.1.7 Grammar and tree representation of acoustical paths

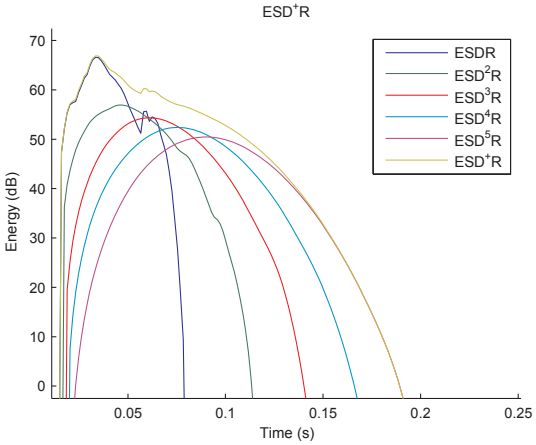
In this section we present two classification methods commonly used in computer graphics to represent the history of a path. These notations have also been used in acoustics by *e.g.* *Dalenbäck* [1996]. These methods will be used later to analyze the pros and the cons of each propagation algorithm. The first method is a classification of algorithms with a grammar that represents the history of all ray reflection in a virtual scene. The second representation is an extension of the grammar with a graph representation.

Acoustical path grammar

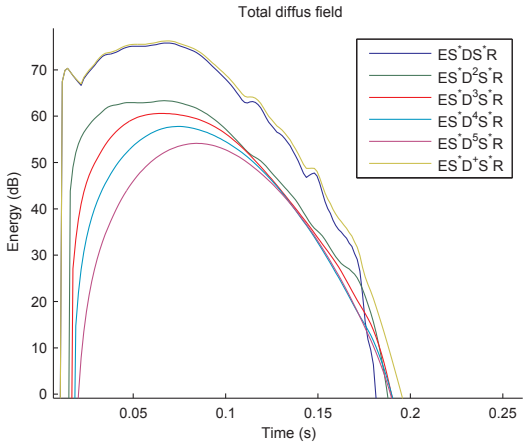
It is usual in the field of computer graphics to represent the history of the travel of a path by a grammar [*Heckbert* [1990]; *Jensen* [2001]; *Veach* [1997]]. This approach has also been used in acoustics [*Dalenbäck* [1996]; *Dalenbäck et al.* [1994]]. Table 2.2 defines a grammar that permits to identify all the types of paths. As an example, the purely diffuse paths produced by radiosity algorithms (see Section 2.1.6) will be of the form $\mathtt{ED^+R}$. Figure 2.7(a) represents pure diffuse energetic transfers between a source and a receiver in the scene described in Appendix B.1.1. The algorithms of deterministic ray tracing



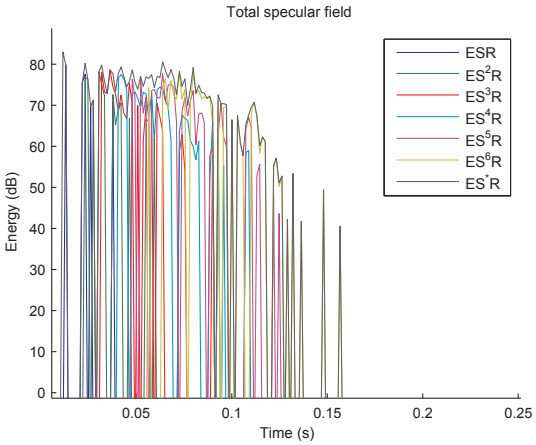
(a) Pure diffuse field.



(b) Pure diffuse field after one specular reflection.



(c) Total diffuse field.



(d) Specular field.

Figure 2.7: Some grammar extractions of the hybrid image source / radiosity algorithm for test scene B.1.1.

E	Emitter
R	Receiver
D	Diffuse reflection
S	Specular reflection
X^+	At least one occurrence of X
X^*	Zero or more occurrences of X
X^N	Exactly N occurrences of X
(D|S)	A reflection either diffuse or specular

Table 2.2: Descriptive grammar of acoustical paths.

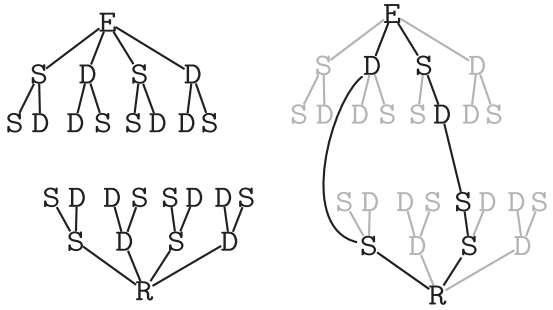


Figure 2.8: Graph representation of a grammar: (left) two trees generated from a source and a receiver, (right) two examples of paths EDSR and ESDSSR.

(see Section 2.1.5) or source image (see Section 2.1.5) will produce pure specular paths (ES^*R). The algorithms based on particles like MCRT will generate all types of paths with diffuse and specular reflections $E(S|D)^*R$.

Graph representation of the paths In order to have a visual representation of all the paths in the virtual scene, the grammar of every path can be drawn on a graph (see Figure 2.8). This method is well suited to the analysis of local phenomena involving a small number of paths. As the number of paths increases, the graph is too large and not readable. It appears non practical in this case.

2.2 Independent processing of specular and diffuse field

Starting from the description of the classical algorithm for geometric acoustic propagation, and the grammar/tree classifications, we present an algorithm that combines image source (see Section 2.1.5) and radiosity (see Section 2.1.6) to determine exhaustively all paths between a source and a receiver in a simple virtual scene.

Our algorithm is based on the decomposition of pure specular reflections $\texttt{ES}^*\texttt{R}$ and diffuse field in a virtual scene. The diffuse field is composed of all paths that have at least one diffuse reflection during their travel, *i.e.*, all paths with grammar $\texttt{ES}^*\texttt{DS}^*\texttt{R}$ or $\texttt{ES}^*\texttt{D}(\texttt{S}|\texttt{D})^*\texttt{DS}^*\texttt{R}$. It is important to note that in these three grammars there is a symmetry between the reflections starting from the source, and those starting from the receiver. This symmetry is used in bidirectional algorithms to determine the origin of the rays — starting from the source, or from the receiver.

2.2.1 Specular paths

We have presented in Section 2.1.5 two algorithms to gather the specular paths in a virtual scene. The algorithm we are presenting aims at giving an exhaustive representation of all paths in virtual scenes with few walls. In order to ensure that no path is omitted, we have selected the image source algorithm. Figure 2.8 shows how two trees are built from the source and the receiver. In order to build the fully specular paths in these trees, the image source algorithm is used twice. The first pass is the classic image source algorithm, images are created from the source up to a given order. The second pass is executed from the receiver. The image receivers are created, they represent, as for image sources, the mirror reflections of the receiver on the walls. At this stage, we have built two trees, one from the source, the other from the receiver, that contain all visible image sources and image receivers up to a given order of reflection. The contribution between a source image and a receiver image[13] is characterized by its arrival time, $t_{\mathcal{E}}$:

$$t_{\mathcal{E}} = \frac{||\mathbf{x}_{\mathcal{E}} - \mathbf{x}_{\mathcal{R}}||}{c} \tag{2.29}$$

and its energy:

$$\Phi_{\mathcal{E}\texttt{S}} = \frac{1}{||\mathbf{x}_{\mathcal{E}} - \mathbf{x}_{\mathcal{R}}||} \prod_{i=1}^{N_{refl}} \vartheta_{\texttt{S},i} \tag{2.30}$$

where $\mathbf{x}_{\mathcal{E}}$ is image source position, $\mathbf{x}_{\mathcal{R}}$, image receiver position and c, the celerity of the sound wave in the medium, N_{refl}, the number of reflections and $\vartheta_{\texttt{S},i}$, the specular reflectance of the ith reflection.

2.2.2 Diffuse paths

In our approach, the denomination *diffuse path* is not restricted to pure diffuse paths $\texttt{ED}^+\texttt{R}$. It includes all paths with at least one diffuse reflection. These paths are separated into three categories, each of them having a specific treatment:

- $\texttt{ES}^*\texttt{DS}^*\texttt{R}$ are the paths containing only one diffuse reflection between two image source/receiver,
- $\texttt{ES}^*\texttt{D}^+\texttt{S}^*\texttt{R}$ are the paths containing more than one diffuse reflection between two image source/receiver,
- $\texttt{ES}^*\texttt{D}(\texttt{S}|\texttt{D})^*\texttt{DS}^*\texttt{R}$ are the paths that may contain specular paths between two diffuse reflections.

[13] Source and receivers are considered as image sources and image receivers. They are usually called zero order image source/receiver.

In the case of paths with only one diffuse reflection $\texttt{ES}^*\texttt{DS}^*\texttt{R}$, it is possible to calculate the diffuse field energy with:

$$\Phi_{\texttt{S}^*\texttt{DS}^*} = F_{\mathcal{E}\to i} * F_{\mathcal{R}\to i} \tag{2.31}$$

where $F_{\mathcal{E}\to i}$ is the form factor between the image source and the diffuse wall, and $F_{\mathcal{R}\to i}$, the form factor between the image receiver and the diffusing surface. $*$ is the convolution operator. As the form factor in acoustics is a mono-dimensional signal depending on time, the convolution is form factor is identical to the convolution of signals defined in Section 1.2.5.

For path containing more than one diffuse reflection ($\texttt{ES}^*\texttt{D}^*\texttt{S}^*\texttt{R}$), a new convolution is applied to each radiative exchange $F_{i\to i+1}$ between walls. The resulting diffuse field energy is:

$$\Phi_{\texttt{S}^*\texttt{D}+\texttt{S}^*} = \left(F_{\mathcal{E}\to i} * F_{\mathcal{R}\to i+N_{diff}} \prod^{N_{diff}-1} F_{i\to i+1} \right) \prod_{i=1}^{N_{diff}} (\vartheta_{\texttt{D},i}) \prod_{i=1}^{N_{spec}} (\vartheta_{\texttt{S},i}) \tag{2.32}$$

where $\prod^{N_{diff}-1} F_{i\to i+1}$ represents the convolution product between each diffuse wall encountered by the path, $i + N_{diff}$ is the index of the last diffuse wall. N_{diff} and N_{spec} are respectively the number of diffuse and specular reflections of the path.

The paths that represent specular reflections between two diffuse reflections $\texttt{ES}^*\texttt{D}(\texttt{S}|\texttt{D})^*\texttt{DS}^*\texttt{R}$ are replaced for simplicity by pure diffuse reflections. As diffuse reflections also carry specular energy, the reflected diffuse energy becomes the total energy (after Equation 2.8):

$$\Phi_{\texttt{D}} = \Phi_{tot} \tag{2.33}$$

2.2.3 Reflection graph algorithm

Starting from the graph representation of the propagation of the sound wave (see Section 2.1.7), the image source / image receiver algorithm, and the diffuse path generation, we propose an algorithm to collect exhaustively all paths between a sound source and a receiver in a virtual scene. As the exhaustive method involves a number of paths exponentially growing with the number of walls and the order of reflections, this algorithm is only applied to simple scenes — *e.g.* scene B.1.1 , with reflection orders up to seven. We call this method the reflection graph, that should not be confused with *Stavrakis et al.* [2008] reverberation graph algorithm. This algorithm is based on a recursive travel across the two specular reflection graphs presented in Section 2.2.1. We remind that those two trees contain all specular reflections generated from the source and the receiver up to a certain order. For each reflection detected while traveling recursively in the trees, a path collect is made. The tree recursion starts from the sound source. At each specular reflection, the path is propagated specularly ($\texttt{ES}^*$), and diffusely ($\texttt{ES}^*\texttt{D}$) toward all the other patches of the scene. When a diffuse reflection is made starting from the source, a recursion on the receiver tree is made. The specular ($\texttt{DS}^*\texttt{R}$) and diffuse ($\texttt{D}^*\texttt{R}$) reflections are then collected. During a recursion from the receiver, a specular reflection generates all specular ($\texttt{S}^*\texttt{R}$) and diffuse ($\texttt{D}^*\texttt{S}^*\texttt{R}$) paths on every wall of the scene.

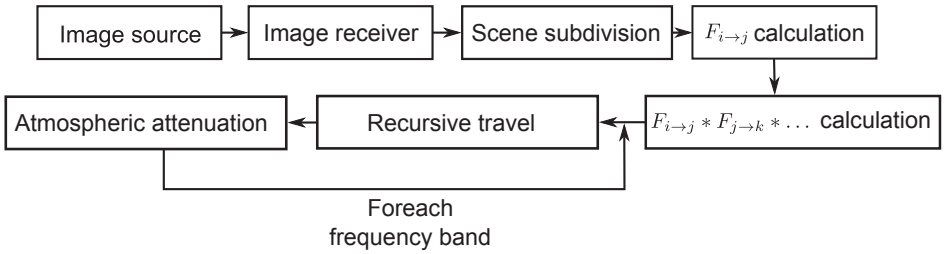


Figure 2.9: Hybrid image source radiosity algorithm.

Finally, a diffuse reflection starting from the receiver collects all reflections (S|R) with diffuse energy (see Equation 2.33). Algorithm 5 shows the pseudo-code of the four recursion functions.

Algorithm 5: The four recursive functions of the reflection graph algorithm.

```
begin ES*{w_{i-1}} // Pure specular from src
    SpecularCollect ()
    foreach w ∈ walls, w ≠ w_{i-1} do
        ES*(w)
        ES*D(w)
begin ES*D{w_{i-1}} // Pure specular ending with a diffuse from src
    DiffuseCollect ()
    foreach w ∈ walls, w ≠ w_{i-1} do
        S*R(w)
        D*S*R(w)
begin S*R{w_{i-1}} // Pure specular from recv
    DiffuseCollect ()
    foreach w ∈ walls, w ≠ w_{i-1} do
        S*R(w)
        D*S*R(w)
begin D*S*R{w_{i-1}} // Pure specular finished by pure diffuses from recv
    DiffuseCollect ()
    foreach w ∈ wall, w ≠ w_{i-1} do
        D*S*R(w)
```

2.3 Implementation of a hybrid source image / radiosity algorithm

Figure 2.9 depicts a summary of the steps that compose the algorithm. It starts with the creation of the image source tree and the image receiver tree as presented in Section 2.2.1. Then the virtual scene is subdivided into triangles to calculate the form factors, F_i, between the image sources and the walls and F_j, between the image receivers and the walls for the radiosity algorithm. The form factors

between the walls, $F_{i\to j}$, are calculated using Monte Carlo integration with the method presented by *Bekaert* [1999] (*cf.* Algorithm 6). Then, all combinations of F_{ij} convolutions are calculated up to the given order of reflection for the simulation — due to memory restrictions, the tests of this algorithm were performed up to six orders of reflections. The end of the processing is repeated once for each frequency band in order to take into account the frequency dependent attenuation of the materials and medium. The frequency dependent processing starts with the exploration of the graph using Algorithm 5. Every time a path is collected, it is added to the global echogram of the simulation. It may also be added to a specific echogram corresponding to the grammar we want to analyze. For instance, if the current path is composed of three specular reflections ($\mathtt{ES^3R}$), and we want to analyze the evolution of specular reflections in the scene, then, the path will be added to the global echogram, to an echogram that gathers the specular paths ($\mathtt{ES^*R}$), and to a third echogram that gathers the third order specular paths ($\mathtt{ES^3R}$).

2.3.1 $F_{\mathcal{E}\to i}$ and $F_{\mathcal{R}\to j}$ calculation

The calculation of the form factors between an image source (or an image receiver) and a wall is performed with a numerical integration on the surface of the patch, as presented in Equation 2.21. This integration is performed with a simple numerical integration. The patches are discretized, and the integration is performed at the middle of the patch — no interpolation is performed. The result of this integration is an echogram that represents the energy exchanges between a punctual entity and a wall of the virtual scene. The echogram has a temporal resolution F_s of 86Hz[14]. Starting from this observation, we found that the distance between two integration points, *i.e.*, the center of the subdivided patches, must be less or equal than d_{sub} with:

$$d_{sub} = \frac{c}{2F_s} \approx 2 \text{ [m]} \tag{2.34}$$

The 1/2 factor is included to respect Shannon's sampling theory. During the image source graph exploration, every path containing at least one diffuse reflection will contain a $F_{\mathcal{E}\to i}$ and a $F_{\mathcal{R}\to j}$ term, i being the index of the first patch and j the index of the last one. Note that in the special case of a unique diffuse reflection $i = j$.

2.3.2 $F_{i\to j}$ calculation

The echograms corresponding to the form factors between two patches of the scene were calculated thanks to a Monte Carlo (MC) integration with an adaptation of the techniques presented by *Bekaert* [1999].

The $F_{i\to j}$ factors satisfy the following relation:

$$F_{i\to j} = \frac{A_i}{A_j} F_{j\to i} \tag{2.35}$$

[14]The temporal resolution of the echogram is based on perceptive parameters of the diffuse sound field presented in Chapter 4.

with A_i and A_j the areas of patches i and j respectively. This property enables us to store only one echogram for each couple of patches in the scene. We also store the areas of both patches in order to apply the correction ratio latter. The stored echogram will be $A_i F_{i\to j}$ or equivalently $A_j F_{j\to i}$.

Algorithm 6 presents the calculation of $F_{i\to j}$ for an enclosed scene, *i.e.*, a scene where the particles stay inside. During the initialization step of the algorithm, each particle is thrown from a sound source of the virtual scene in a random direction. The first intersection performed with a wall $\mathcal{I}_i$ is stored, and the particle is reflected omni-directionally on the hemisphere above the wall intersected. Then, a loop is performed until the calculated $F_{i\to j}$ converges. The next intersection $\mathcal{I}_j$ of the particle is also stored. The two walls intersected, i and j, will be used for the computation of the form factor $F_{i\to j}$. The propagation time between the two intersections $(\mathcal{I}_{j,time} - \mathcal{I}_{i,time})$ gives the bin of the form factor to increment. The form factor is the amount of energy exchanged between the two patches i and j. In order to obtain this information from the previous calculation, we need to store the number of particles N_i emitted from patch i, to establish the ratio between the number of emitted particles N_i, and the number of particles reaching patch j[15]. This correction factor, is applied along with the wall area correction presented in Equation 2.35 as the last step of the algorithm.

In simple scenes like the one presented in Appendix B.1.1, a number of tries N_{try} of the order of 100.000 particles is sufficient to have a low variance on the $F_{i\to j}$ factors.

Algorithm 6: $F_{i\to j}$ calculation using particles.

```
/* Initialization                                                    */
Particle P
x_P ← E_pos
d_P ← UniformSampleSphere()
I_i = GetNearestIntersection()
d_P ← UniformSampleHemisphere()
/* Processing                                                        */
foreach N_try do
    I_j = GetNearestIntersection()
    d_P ← UniformSampleHemisphere()
    F_{i→j}[I_{j,time} − I_{i,time}] ← F_{i→j}[I_{j,time} − I_{i,time}] + 1
    N_i ← N_i + 1
    I_i ← I_j
/* Apply correction factor                                           */
foreach F_{i→j} do
    A_i F_{i→j} ← (A_i F_{i→j}) / N_i
```

$F_{i\to j}$ convolution

As the algorithm operates on simple scenes, the number of $F_{i\to j}$ factors is small, and can be pre-calculated and stored for each patch couple. It is also important to note that the $F_{i\to j}$ factors are

[15] This information is stored in $F_{i\to j}$ during the processing.

independent of the position of the source and receivers. On the other side, F_i and F_j have to be calculated on the fly as they are relative to the position of the source image and the receiver image. We have seen in Equation 2.32 that a diffuse path of type $\texttt{ES}^*\texttt{D}^+\texttt{S}^*\texttt{R}$ may contain one or more form factors that are convolved ($\prod^{N_{diff}-1} F_{i\to i+1}$). This convolution is also independent of the position of the source and the receiver. It is also pre-calculated for each combination of patches up to the simulation order. In order to reduce the number of echograms calculated and stored, we use the commutativity property of the convolution :

$$F_{i\to j} * F_{j\to k} = F_{j\to k} * F_{i\to j} \tag{2.36}$$

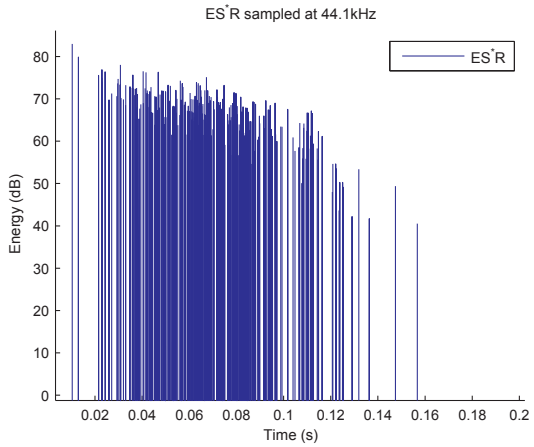
2.3.3 Collecting diffuse and specular paths

The aim of our analysis algorithm is to extract and compare algorithms that have different characteristics, and that can be described or parameterized using a grammar. We have presented in Algorithm 5 a recursive method to collect exhaustively all the paths of the scene. We will describe here the implementation of the specular and diffuse collects. The results of the simulations will be presented in the form of echograms. In order to have a coherent analysis of the specular and diffuse fields, the echograms must have the same resolution. Usually, for auralization, we sample echogram at 44.1 kHz, but as the diffuse field echograms are sampled at 86 Hz, we choose to sample all echograms at 86 Hz. Figure 2.10 shows three integrated echograms for the specular paths; the first one has a sampling frequency of 44.1 kHz, the second has a sampling frequency of 689 Hz (= 44100 / 64), and the last one has a sampling frequency of 86 Hz (= 44100 / 512). Tests have been performed with various sampling frequencies from 1378 Hz (= 44100 / 32) to 10 Hz (= 44100 / 4096). They did not show better results above 86 Hz, as the diffuse response in real room has a low level of details. Below 86 Hz, the curve becomes too smooth, and thus details are lost. In this thesis, the sampling frequency of 86 Hz is kept for the diffuse echograms.

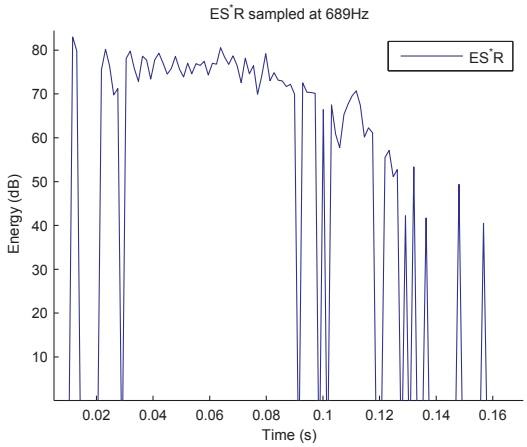
The specular collect phase is made every time a pure specular path ($\texttt{ES}^*\texttt{R}$) is found during the graph exploration. The time of arrival in the echogram and the energy collected are given by Equations 2.29 and 2.30. Here, the arrival time is punctual, as the echogram is discretized, the energy is collected in the nearest echogram bin. As the energy is spread in the bin, it also has to be divided by the sampling frequency of the echogram F_S :

$$\Phi_{bin} = \frac{\Phi_\texttt{S}}{F_s} \tag{2.37}$$

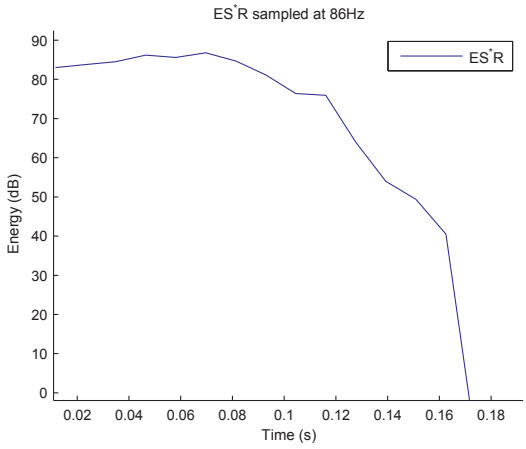
For the diffuse paths collection, the energy exchanged between the source and the receiver is spread over time. Equation 2.32 shows that the resulting echogram is the convolution of one echogram for the exchange between an image source and the first diffuse wall $F_{\mathcal{E}\to i}$, one convolution for each diffuse wall found during the travel $F_{i\to i+1}$ and one convolution for the exchange between the image receiver and the last wall found $F_{\mathcal{R}\to j}$. Figure 2.12 shows the geometry of the exchanges for a path of type ESDSR (left), and a path of type ESDDSR (right). Figure 2.11 shows the two echograms generated from the



(a) Specular echogram sampled at 44.1 kHz.



(b) Specular echogram sampled at 689 Hz.



(c) Specular echogram sampled at 86 Hz.

Figure 2.10: Three different integration steps on a specular echogram.

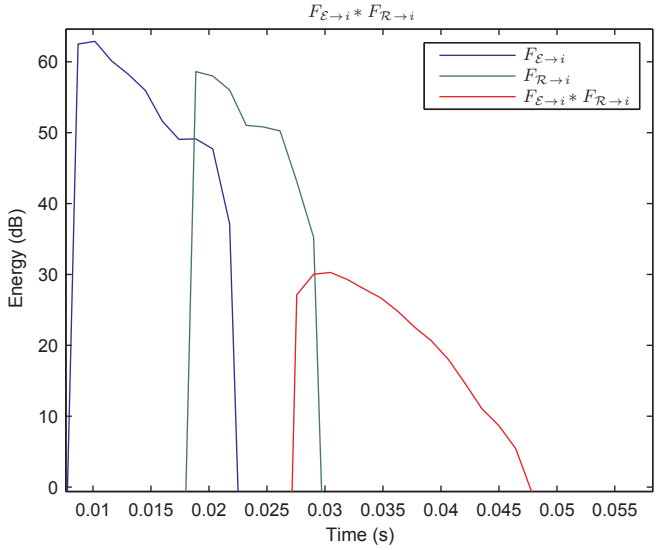


Figure 2.11: The result of the convolution of two form factors $F_{\mathcal{E}\to i}$ between a source image and a wall, and $F_{\mathcal{R}\to i}$ between the same wall and a receiver image.

source and the receiver. It also shows the result of the convolution of these two echograms.

During the exploration of the graph, a global echogram is created that gathers all paths that have a contribution between the source and the receiver. Paths are also collected while they match with a given grammar, in order to analyze the contributions of various algorithms (characterized by their grammar). Examples of realizations of this algorithm are given in Figure 2.7.

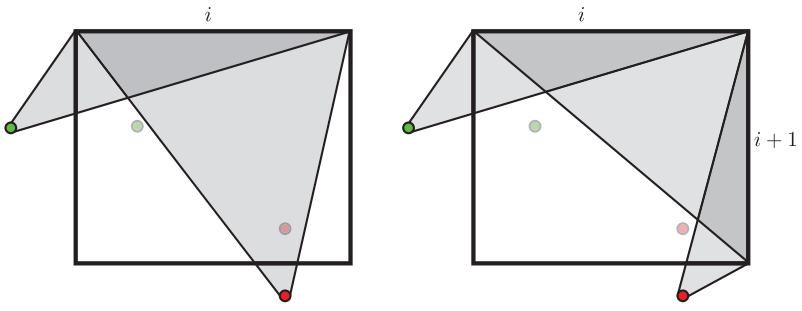


Figure 2.12: Evaluation of the form factors : (left) exchanges between an image source, a wall and an image receiver $F_{\mathcal{E}\to i} * F_{\mathcal{R}\to i}$, (right) exchanges between an image source, two walls and an image receiver $F_{\mathcal{E}\to i} * F_{i\to i+1} * F_{\mathcal{R}\to i+1}$.

2.4 Spatial coherence of the sound field in rooms : the rendering hierarchy

Starting from the hybrid image source / radiosity algorithm, we have extracted two characteristics of the propagation algorithms: the depth of reflections, and the extraction of purely specular, and diffuse fields. We decided to use this algorithm in order to provide a parameter to tune the real-time algorithms. The *spatial coherence* parameter gives an estimation of how much the purely specular and the diffuse sound fields vary in an enclosed space. Bigger variations of the sound field will involve more computational resources allocated to the corresponding reflection order of the algorithm. The simulation with the hybrid algorithm was performed between each couple of source/receiver of the third round robin on room acoustics (see Appendix B.1.1). Figure 2.13 shows the realizations of the simulation of four different grammars ESR, ES^{5}R, EDR, ED^{5}R. In order to have equivalent echograms for specular and diffuse fields, the specular echogram is integrated over a period of 11.6ms — that is equivalent to diffuse field sampling of 86 Hz. Figure 2.14 shows the standard deviation ρ for the realizations of echograms over the six positions of the source and receiver. The aim of this analysis is to establish a hierarchy between the propagation algorithms, and their reflection depth. The criterion we kept for the spatial coherence is the maximum of standard deviation between the realizations. As expected, we observed that the diffuse field is more coherent than the specular field, and the coherence grows with the number of reflections. But, one of the major information to extract from Figure 2.14 shows that the spatial coherence of specular paths of order five is equivalent to the coherence of diffuse paths of order one in the test scene (the value of the spatial coherence is indicated above each curve). In the realization of a real-time propagation algorithm, this means that the budget[16] allocated to the specular paths of order five and diffuse path of order one must be of the same order.

The main drawback of this method is that all local effects of the rays are hidden as the energy of the ray is summed. The energy of the rays is calculated from their history, *i.e.*, the distance between the source and the receiver and the materials of the walls intersected. However, this method produces a rough view of how the reflections, whether they are specular or diffuse, have an influence on the impulse response. And, particularly how this response is modified when the source or the receiver moves.

* * *

In this section, we have developed an exhaustive method to enumerate all paths between a source and a receiver in a virtual scene. The ranking of the paths according to a graph of a grammar is used to characterize the algorithm — the type of algorithm or its parameters. The methods of dynamic sound rendering are in general constructed around an optimized algorithm dedicated to reproduce as many propagation information as possible within the allocated time and computational resources. The analysis provides a first step to the conception of these algorithms; it organizes in a hierarchy the propagation steps according to the spatial coherence criterion, *i.e.*, algorithms with lower coherence will be allocated less budget. So, it is possible starting from these results to parameterize real-time algorithms in simple

[16] Computational resources.

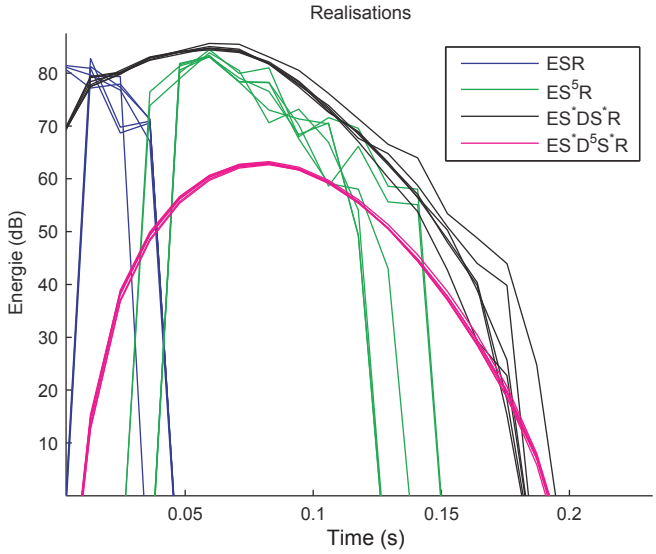


Figure 2.13: Extraction of the grammar for six realizations of the hybrid image source radiosity algorithm for the six couples of points/receivers of scene B.1.1.

scenes. Note that, the exhaustive approach has its limits. It is impossible to make exhaustive simulations on the test scene of Appendix B.1.1 up to seven orders of reflections due to restrictions on calculation time and memory storage.

In order to reach higher reflection orders, and thus allocate the budget to the specular and diffuse algorithm up to orders 100 or more, a different algorithm should be used. We have thought about a variant of Monte Carlo particle tracing where the purely specular and diffuse particles are collected independently. However, this algorithm has not been implemented.

The exhaustive approach allows to characterize independently each path, and to gather them according to their grammar. In the next section, we describe the implementation of two real-time algorithms dedicated to specular and diffuse fields respectively.

2.5 Implementation of sound propagation with specular and diffuse reflection algorithms for real-time auralization

Starting from the observations of the previous chapter, we have implemented two propagation algorithms suited for real-time rendering. As explained earlier, the exhaustive approach is computationally too expensive, so it is not applicable to real-time rendering. The two algorithms that were implemented are derived respectively from deterministic ray tracing for specular contributions, and Monte Carlo particle

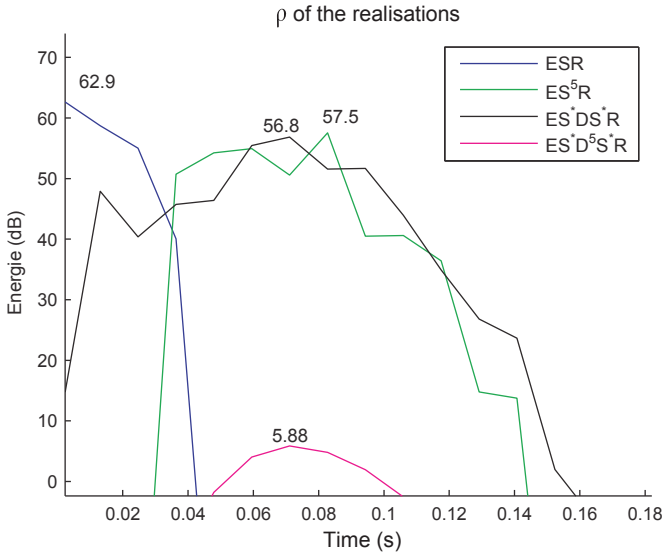


Figure 2.14: Standard deviations between the observations of Figure 2.13. The maximal values of each curve are used as coherence criterion.

tracing for the diffuse part of the echogram[17]. The idea behind the separation of pure specular and diffuse field, is to have algorithms operating in parallel on recent computer architectures. The priorities allocated to each algorithm, can then be established with the coherence criterion defined in Section 2.4. Another reason for the separation between the specular and diffuse algorithms, is that the perceptive mechanism associated with the rendering of each phenomenon can be studied separately (see Chapter 4).

2.5.1 Deterministic ray tracing algorithm with growing sphere

Starting from the simple classical DRT Algorithm 2, the analysis made on the receivers (see Section 2.1.4), and the fact that our algorithm should be prioritized depending on the depth of reflections, we present below a new algorithm that collects efficiently specular rays.

This algorithm is composed of four main elements:

- The rays R are the elements propagated in the virtual scene. A ray is defined by its position, $\mathbf{x_R}$, and its direction, $\mathbf{d_R}$, at a given time in the simulation. The energy carried by the ray is Φ_R, and its travel distance is d_R. As the ray propagates through the scene, it keeps the history of its travel, $R_{history}$. The history is the ordered list of all walls intersected during its travel. If two rays have the same history as they reach the receiver — this means that they are issued from the same image source — and thus only one is kept.

[17] *Rays* and *particles* are expressed in the sense of definitions 2.1 and 2.2.

- The rays originate from a sound emitter, $\mathcal{E}$, that is characterized by its position, $\mathbf{x}_{\mathcal{E}}$, and the energy it emits, $\phi_{\mathcal{E}}$. A directivity function, $\alpha_{\mathcal{E}}$, can also be associated with the source.

- While a ray propagates through the scene, it creates intersections, $\mathcal{I}$, every time it interacts with a wall. An intersection is characterized by its position, $\mathbf{x}_{\mathcal{I}}$, the wall that was hit, $\mathcal{I}_{wall}$, the specular absorption of the wall, $\alpha_{\mathcal{I}\mathtt{s}}$, and the distance to the previous intersection, $d_{\mathcal{I}}$.

- The last structure is the receiver, it has the shape of a sphere and it collects the rays that cross the sphere during their travel. The receiver $\mathcal{R}$ is characterized by its position $\mathbf{x}_{\mathcal{R}}$ and its radius $r_{\mathcal{R}}$.

This algorithm starts with the generation of rays in the virtual scene. The generation uses an omnidirectional uniform sphere sampling. Then, the ray travels through the virtual scene. The main difference with classical deterministic algorithms is that the rays do not travel until they are absorbed. Instead, in our approach, we launch all the rays, and, as they interact with a wall, their reflection direction is calculated, and the intersection with the receiver is checked. With this arrangement of the rays propagation, it is easier to control the depth of propagation of the algorithm as the next step of the algorithm is identical: the rays will be propagated from their current location (the position of the last intersection).

The main advantage of the current algorithm is that it can be stopped every time an algorithm with higher priority needs computational resources, at the cost of an increase of memory (the rays have to be all stored).

We have also implemented the concept of growing sphere for this algorithm. According to Equation 2.10, the radius of the receiver in free field depends on the maximum time of the echogram, the speed of sound, and the number of particles. Let t_{sup} be the maximal time a ray can travel in the virtual scene between two reflections[18]. At each reflection order o of the algorithm, we can update the radius of the receiver as:

$$r_{\mathcal{R}} = o\frac{c\ t_{sup}}{\sqrt{N_{rays}}} \tag{2.38}$$

The main advantage with a deterministic ray tracing algorithm is that the frequency dependence of the material has no influence on the direction of reflection of a ray. Thus, we can perform only one ray shooting for all frequency bands. If we consider frequency dependent materials, the energy of the ray, Φ_R, becomes a spectrum, and contains the number of values used for the simulation.[19]

The incremental deterministic ray tracing with growing sphere is presented in Algorithm 7. It is important to note that this algorithm does not consider the direct sound between the emitter and the receiver path, instead, the visibility between the source and the receiver is tested directly.

[18] t_{sup} can be approximated by a first step of ray tracing using very few rays. It can be updated during the algorithm if one of the rays has a longer travel time than t_{sup}.

[19] In our implementation, we use six octave bands values from 125 Hz to 4 kHz.

Algorithm 7: Incremental deterministic ray tracing with growing sphere algorithm.

```
/* Initialization                                                    */
foreach Ray R do
    x_R ← x_E
    d_R ← UniformSampleSphere()
    R_d ← 0
    Φ_R ← Φ_E
/* Processing                                                        */
while NextStep? do
    UpdateReceiverRadius ()
    foreach Ray R do
        I = GetNearestIntersection()
        if I_type is Wall then
            x_R ← x_I
            d_R ← SpecularReflection(I)
            Φ_R ← Φ_R · α_I
            d_R ← d_R + d_I
            R_history ← R_history|I
        else if I is ∞ then
            continue
        if IntersectReceiver () then
            Collect (R)
```

2.5.2 Incremental Monte Carlo particle tracing algorithm

Starting from the deterministic algorithm of Section 2.5.1, the classical MCRT algorithm, and the analysis of Section 2.4, we present an incremental algorithm dedicated to the reproduction of diffuse sound field in enclosed spaces. This algorithm shares some characteristics with the incremental deterministic algorithm. The intersections, $\mathcal{I}$, and emitter, $\mathcal{E}$, have the same properties. In this algorithm, these are no longer rays that are propagated, but particles, P. A particle has a position, $\mathbf{x_P}$, and a direction, $\mathbf{d_P}$, like a ray, but a particle does not travel with the total energy emitted by the source. Instead, the energy is distributed uniformly on all the particles of the simulation. The energy carried by a particle is defined in Equation 2.22. Another important difference is that a particle does not conserve the history of its travel.

The collect of the particles is made by a spherical receiver. Every time a particle crosses the receiver it is collected and stored in the echogram. The receiver, $\mathcal{R}$, is characterized by its position, $\mathbf{x}_\mathcal{R}$, and its radius, $r_\mathcal{R}$.

As the MCRT algorithm is designed for both specular and diffuse field, it is important to omit all pure specular paths ($\mathtt{ES^*R}$) reaching the receiver. The specular paths have to be discarded because they are already collected by the deterministic algorithm. With this method, we can collect all paths containing at least one diffuse reflection.

The particularity of this algorithm is that it can be paused at every reflection order of the particles

in the similar way as the deterministic algorithm. The particles are first emitted from the sound source. They then travel through the virtual scene until they intersect a patch. The frequency dependent reflections are performed using *Embrechts* [2000] splitting coefficient method described in Section 2.1.6. As the splitting coefficient, β_P, is linked to the history of a particle, it is also part of the particle object. The corrective attenuation coefficient, C_P, (see Equation 2.26), is also stored in the particle. Once they reach a wall, the particles are reflected either diffusely or specularly. The position and direction of the particles are stored, and the intersection test with the receiver is performed for the collect.

Algorithm 8: Incremental Monte Carlo particle tracing.

```
/* Initialization                                                    */
foreach Particle P do
    x_P ← x_E
    d_P ← UniformSampleSphere()
    d_P ← 0
    Φ_P ← Φ_E / N_part
/* Processing                                                        */
while NextStep? do
    foreach Particle P do
        I = GetNearestIntersection()
        if I_type is Wall then
            β_P = max_f{C_P I_S} / (max_f{C_P I_S} max_f{C_P (1 − I_S)})
            x_P ← x_I
            Φ_P ← Φ_R · (1 − α_I)
            ξ = Rand[0..1]
            if ξ > β_P then /* Specular                              */
                d_P ← SpecularReflection(I)
                C_P ← C_P · (1−s)/β_P
            else /* Diffuse                                          */
                d_P ← DiffseReflection(I)
                C_P ← C_P · s/β_P
        else if I is ∞ then
            continue
        if IntersectReceiver (P) & NonPureSpecular (P) then
            Φ_P ← Φ_P · C_P
            Collect (P)
```

As this algorithm concentrates on diffuse field, it is not necessary to conserve the directional information during the collect. So, the resulting echogram is independent of the orientation of the receiver.

2.6 Conclusion

In this chapter we have presented some of the algorithms commonly used for the sound wave propagation. Starting from the analysis of these algorithms, we have developed and hybrid algorithm based on image sources and radiosity to enumerate exhaustively the diffuse and specular sound path in an enclosed space. Starting from the independent analysis of the diffuse and specular reflections, we presented a criterion that prioritizes the algorithms of propagation. This criterion is based on the spatial coherence of the sound field at different orders of specular and diffuse reflections. Finally, we presented the two propagation algorithms that we implemented to produce real time propagation of the diffuse and specular paths. The details about the implementation of these algorithms and the global parallel structure is presented in Chapter 5. The validation of both the quality of rendering and execution time is presented in Chapter 6. The next chapter will present the Digital Signal Processing (DSP) methods used of the auralization for the information generated by the two real-time algorithms.

3

Auralization of the pressure sound field

In this chapter we present the auralization of sound based on Digital Signal Processing (DSP) operations. Starting from the propagation algorithms described in the previous chapter, we present in Section 3.1 various auralization methods from the literature. We first present the binaural auralization of pure specular paths (see Section 3.2). In Section 3.3, we present a convolution algorithm implemented on Graphical Processing Unit (GPU) to perform the auralization of diffuse sound field generated by Monte Carlo Ray Tracing (MCRT) algorithms. In Section 3.4, we discuss the combination of specular and diffuse sound fields in room acoustics, and compare it to the classical room acoustics early/late reflection model presented in Section 1.4.1. Finally, in Section 3.5, we present a decomposition method for Room Impulse Response (RIR) modeling called Specular/Cluster/Diffuse (SCD).

Contents

Starting from the geometric information provided during the propagation stage, we report in this section different methods used to auralize the sound field existing in a virtual scene. In a first part, we present known methods from the literature for auralization. We then report the modifications we propose in order to implement a real-time auralization of both specular and diffuse field in complex environments.

3.1 State of the art

In this section we report the DSP operations applied to an anechoic signal to produce audio rendering. There are two families of auralization techniques. The first one is based on the independent audio rendering of paths. It is limited to a small number of paths (around 100 on current computers), but it allows to have a precise management of each path. Complex effects like Doppler shift (see Section 1.2.8) can be implemented with this technique. The second family is based on the auralization of the entire sound field with convolution methods (see Section 1.2.5). With a modern implementation of convolution, it is possible to reproduce a full Room Impulse Response (RIR), but the individual path information is lost, only the delayed sum of their contribution is kept.

3.1.1 Auralization of a single specular ray

A single specular ray traveling in the virtual scene has the following history pattern (given by the RARE, *cf.* Section 1.1.5). It is first emitted by the sound source which is characterized by its emission pattern, $\alpha_{\mathcal{E}}$. Then it travels through the scene. At every intersection with a wall of the virtual scene, the ray gets attenuated by the specular BRDF of the material, $f_{r\mathtt{S}}$. The ray gets also delayed by the propagation operator, S_r. Finally, as the ray reaches the receiver, it gets attenuated by the directivity function of the receiver, $\alpha_{\mathcal{R}}$. More complex directivity functions can be implemented when dealing for instance with binaural rendering, these methods will be developed in Section 3.1.3. It is important to note that all the attenuations applied to a ray are in general frequency dependent, and may not be reduced to a single gain, *i.e.*, an amplification factor.

From a DSP point of view, ray corresponds to a single channel between a source and a receiver. The operations applied to a ray are frequency dependent gains and delays. They can be decomposed as:

$$s_{\mathtt{E}}[n] = s_{\mathcal{E}}[n] * h_{\alpha_{\mathcal{E}}}[n] \tag{3.1}$$

$s_{\mathtt{E}}$ represents the same radiated pressure including the effects of the frequency dependent radiation pattern, ($h_{\alpha_{\mathcal{E}}}$). $s_{\mathcal{E}}$ represents the source signal, it must be an anechoic sound.

As the ray travels through the virtual scene, it gets filtered by the materials it encounters, and delayed to take into account the propagation time. The signal, $s_{\mathtt{ES}^*}$, after N_{refl} reflections is

$$s_{\mathtt{ES}^*}[n] = s_{\mathtt{E}}[n] \prod_{i=1}^{N_{refl}+1} h_{\vartheta_{\mathtt{S},i}}[n] * \delta\left[n - \frac{|\mathbf{x_i} - \mathbf{x_{i-1}}|}{c}\right] \tag{3.2}$$

with $\prod$, the convolution product of the material filters, $h_{\vartheta_{\mathrm{S},i}}$, propagation delays, δ, and $|\mathbf{x_i} - \mathbf{x_{i-1}}|$, the distance between two successive intersections along a path[1]. Finally, when the ray is captured by the receiver, the attenuation linked to the detection function is applied to the signal.

$$s_{\mathrm{ES}^*\mathrm{R}}[n] = s_{\mathcal{R}}[n] = s_{\mathrm{ES}^*}[n] * h_{\alpha_{\mathcal{R}}}[n] \tag{3.3}$$

The global propagation of a ray in terms of DSP operations can thus be expressed as:

$$s_{\mathcal{R}}[n] = s_{\mathcal{E}}[n] * h_{\alpha_{\mathcal{E}}}[n] * h_{\alpha_{\mathcal{R}}}[n] \prod_{i=1}^{N_{Refl}+1} h_{\vartheta_{\mathrm{S}}}[n] * \delta\left[n - \frac{|\mathbf{x_i} - \mathbf{x_{i-1}}|}{c}\right] \tag{3.4}$$

Using the linearity properties of the filters and delay, this leads to

$$s_{\mathcal{R}}[n] = s_{\mathcal{E}}[n] * \delta\left[n - \frac{R_d}{c}\right] * \underbrace{\left[h_{\alpha_{\mathcal{E}}}[n] * h_{\alpha_{\mathcal{R}}}[n] \prod_{i=1}^{N_{Refl}+1} h_{\vartheta_{\mathrm{S}}}[n]\right]}_{h_{\mathrm{ES}^*\mathrm{D}}[n]} \tag{3.5}$$

with R_d, the total distance traveled by a ray. This last equation shows that the propagation of a ray can be implemented in DSP using one delay that represents the total travel time, and one filter being the convolution of all filters encountered during the travel. As the convolution of filters is a heavy process, this step is simplified using octave band filtering (see Section 1.3.2). The convolution of filters corresponds to a multiplication in the frequency domain. So the octave band attenuation gains of all filters are multiplied to build the reflection filter of a specular ray $h_{\mathrm{ES}^*\mathrm{R}}$. To be complete, the definition of the filter should also include the convolution by the air absorption $\mathcal{H}^{\alpha_m}$ defined in Section 1.1.5. The frequency dependence is managed using octave band decomposition of the spectrum of the ray.

3.1.2 Auralization of the specular paths

In the current section and the following one, we present the works conducted by *Deille et al.* [2006a,b] on the rendering of specular paths, *i.e.*, the specular paths given by the specular reflection algorithm. As seen before, the linearity of signal processing operators permits the factorization of some operations. When more than one ray travels in the virtual scene, one DSP channel is created per propagated ray. All of them share the same origin (the signal attached to the emitter). Figure 3.1 shows the processing of a path before it gets spatialized. The separation of the spatialization step is a good way to produce a modular virtual application, as the modification of the system (monophonic, stereophonic, binaural, ... see Section 1.3.4) only changes the part of the algorithm dedicated to the spatialization, not the whole signal processing algorithm. We present in the next section how binaural spatialization is performed.

The first step of the auralization is the decomposition of the signal into octave bands, as described in Section 1.2.6. The filtering of the signal is performed using an octave band filter bank, the octave band

[1] $\mathbf{x_0}$ is the position of the source, and $\mathbf{x_{N_{refl}+1}}$ is the position of the receiver.

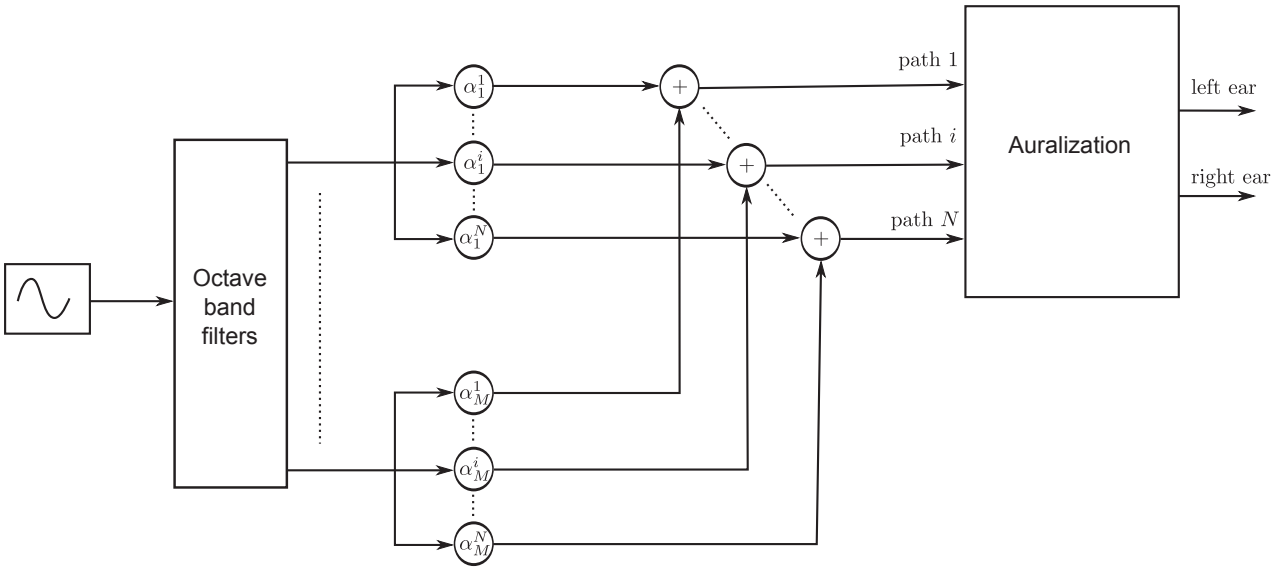


Figure 3.1: DSP propagation algorithm, after *Deille et al.* [2006a].

signals, $s^i_{\mathcal{E}}$, are multiplied by gains, α^i_j, that correspond to the attenuation of the central frequency of the octave band — $i \in [1 \cdots N_{bands}]$ represents the octave band number, and $j \in [1 \cdots N_{rays}]$ the number of rays propagated. In Equation 3.5, we presented the filter $h_{\texttt{ES*R}}$ that represents the full attenuation of a path during its travel. The α^i_j are calculated as the square root of Sabine coefficient for a given material and a given frequency band. For *Deille et al.* [2006a], the filter does not include the receiver filtering — as it is part of the spatialization. So, the α^i_j represent the values of the filter $h_{\vartheta_s,k}$,

$$h_{\texttt{ES*}}[n] = \frac{1}{d_R^2}\mathcal{H}^{\alpha_m}\ h_{\alpha_\mathcal{E}}[n] * \prod_{k=1}^{N_{Refl}+1} h_{\vartheta_s,k}[n] \tag{3.6}$$

d_R^2, represents the geometric divergence, $\mathcal{H}^{\alpha_m}$, the air absorption, $h_{\alpha_\mathcal{E}}$, the emission pattern and $h_{\vartheta_s,k}$, the impulse response associated with the reflection on the material for kth order of reflection. Finally, the octave band signals are summed to recreate the filtered paths. It is important to note that at this stage, no delay was applied. *Deille et al.* [2006a] transfered the delays to the spatialization step to factorize the delays of propagation and the ITD. A pure propagation algorithm would include the propagation delay in this structure.

3.1.3 Digital Signal Processing (DSP) of binaural rendering

Starting from the description of binaural rendering (see Section 1.3.4), we present here the basic signal processing operations necessary for the rendering of a sound spatialized with HRTF. The DSP block diagram is shown in Figure 3.2. In a first step, the signal is split into two channels, for the left and right ear. A delay is applied to these channels to take into account the propagation time of the sound

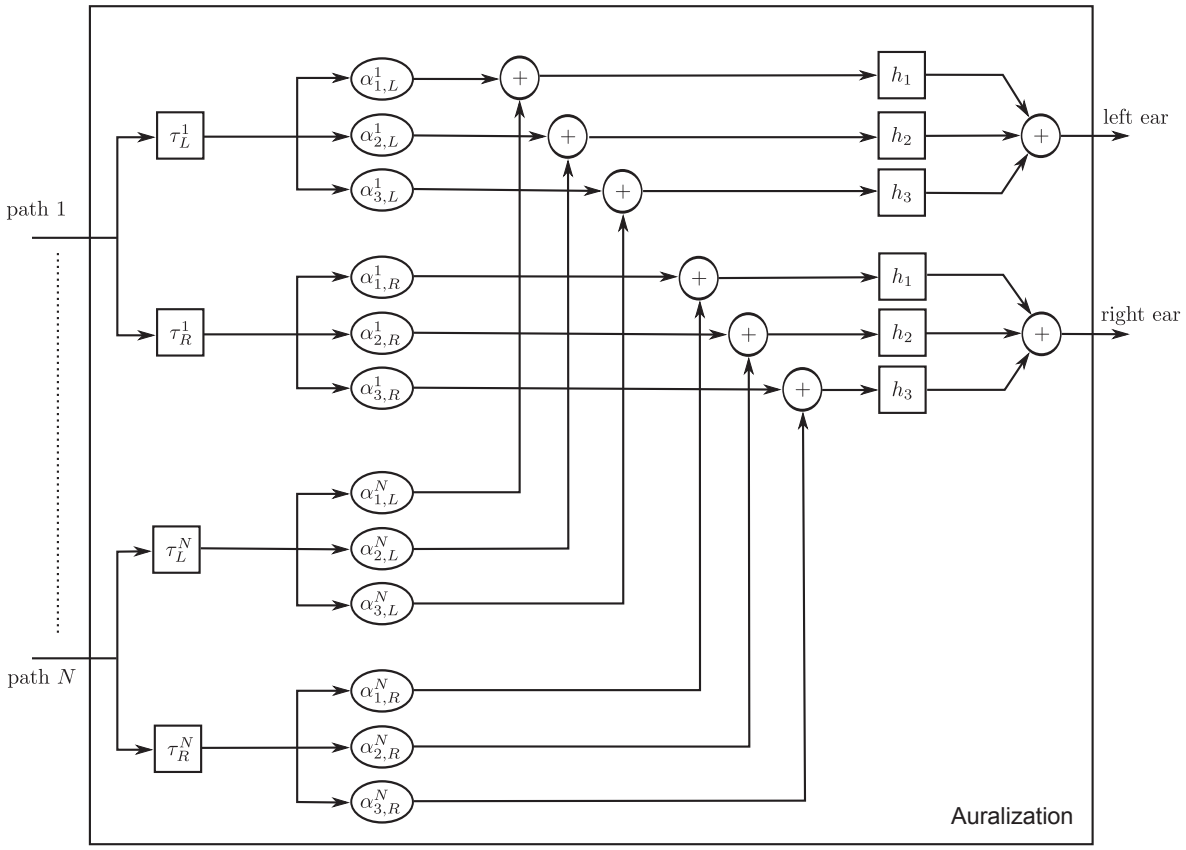


Figure 3.2: DSP binaural auralization algorithm, after *Deille et al.* [2006a].

between the two ears. Then, each path is filtered by the Head Related Transfer Functions (HRTF) filter corresponding to the incoming direction of the path. The HRTF filter models the effect of the interactions with the listener's head. These interactions are complex, thus, octave band filtering is not sufficient to model this phenomenon. So, filtering must be performed using complex filters parameterized by the direction of arrival of the ray on the listener. The implementation of the filters will be presented in the next sections.

3.1.4 Efficient Independent Component Analysis (ICA) decomposition of the Head Related Transfer Functions (HRTF)

The method presented in the previous section is too expensive to provide the auralization of a great number of paths in real-time. *Emerit et al.* [1995] proposed a method based on the ICA decomposition of HRTF to reduce the number of filters used. This method was implemented by *Deille et al.* [2006a,b] to factorize the rendering of many paths using a combination of ICA filters. Figure 3.2 shows the DSP block diagram of the binaural auralization algorithm.

The first step of the binaural processing is to apply the Interaural Time Difference (ITD). A path entering the spatialization step is split into two channels for right and left ear, each of them having a delay, τ_L^i and τ_R^i. In order to have an accurate rendering of the position of the paths, and to have smooth transitions when paths are moving, fractional delays are used at this stage. We have seen earlier that the propagation module presented by *Deille et al.* [2006a] does not apply the propagation delay. Instead, the delay is included in the spatialization, so, the delay of a path corresponds to the delay traveled in the virtual scene, plus the ITD.

The ICA decomposition of the HRTF creates a set of filters for the right and left ear. A set of weighting coefficients $\alpha_{j,L}^i$ and $\alpha_{j,R}^i$ are associated for every position of incoming rays[2]. With this method, every filtered path is only multiplied by one gain per filter, h_j. The number of filters is thus constant, and independent of the number of paths to auralize. That makes this method efficient. Figure 3.2 shows the example of N_{rays} spatialized with a bank of three filters per channel (left and right).

3.1.5 Auralization of diffuse and specular fields by convolution

Another method commonly used for auralization is the convolution of the anechoic source sound by the RIR. This is a FIR filtering operation (see Section 1.2.6), where the filter kernel is the RIR. This filtering replaces all previous filtering, gain and delay operations by single filtering operation.

Specular auralization by convolution

The RIR is created by using the information gathered during the propagation phase. To produce the specular information, the DSP algorithms presented in the previous section can be used. If we recall from Section 1.2.5 that the response of a filter corresponds to its output when a Dirac distribution is passed as input, we can use specular auralization algorithms with an impulsive sound to produce the specular part of the RIR. The aim of this method is to process only once the audio processing graph associated with specular reflections. The drawback is that the produced impulse response has to be regenerated every time the source or the receiver moves in the virtual scene, and no dynamic processing such as Doppler shift can be applied.

Rendering of both specular and diffuse reflections by convolution

Another method to reproduce both specular and diffuse reflections was presented by *Kuttruff* [1993]. This rendering method is based on the auralization of an echogram generated by particle tracing. We have seen in the previous chapter that particle tracing permits to generate echograms containing all specular and diffuse paths (`E(S|D)*R`) between a source and a receiver in a virtual scene. The problem with this approach is that the echogram represents the radiative exchanges between a source and a receiver. Thus, the phase information is lost. However, according to *Kuttruff* [1993], the phase of the signal for virtual acoustic auralization is subjectively insignificant. *Kuttruff* [1993] proposes the following

[2]The azimuths and elevation are divided in 100 slices, then, the gains are interpolated between the nearest neighbors.

method to perform auralization of a frequency dependent integrated echogram (see Section 1.2.4). The integrated echogram is *filled* with Poisson-distributed points that have the same energy as the bins of the echograms to auralize. In order to have a smooth transition between the bins of the echogram, the authors use a smoothing function. We present in Section 3.3.1 an algorithm based on the Ebinaure method (*van Maercke and Martin* [1993]) to auralize the diffuse sound field.

The studies presented in this section are mainly those that were used at Centre Scientifique et Technique du Bâtiment (CSTB) during the past decade. The interested reader may refer to *Vorländer* [2008] or *Kleiner et al.* [1993] for general informations on auralization, or *Savioja et al.* [1999] for the methods used in other auralization systems such as DIVA.

3.2 Auralization of pure specular paths, the Binaural Spatialization Algorithm (BSA)

Starting from the auralization algorithm presented by *Deille et al.* [2006a], after *Emerit et al.* [1995], and the specular deterministic ray-tracing algorithm presented in Section 2.5.1, we present here the auralization algorithm we have developed for the rendering of specular paths. This algorithm is designed to produce a binaural auralization of the specular paths. It manages paths linked to both static and dynamic sources/receivers.

3.2.1 Description of the specular auralization process

Figure 3.3 presents the full procedure to produce a binaural auralization system. This system was designed to spatialize specular paths reaching the receiver. As the delay lines are based on fractional delay, Doppler effect (see Section 1.2.8) can be applied to every spatialized path with no audible artifacts.

We have planned to implement these algorithms on GPU, and compare the execution times with CPU implementation. However, the GPU implementation of these algorithms is not yet functional while writing this thesis.

Octave band splitting The first step of the algorithm is the decomposition of an input signal in octave bands. The implementation of *Deille et al.* [2006a] was based on a preprocessing of the signals[3] to extract the octave bands. In our implementation, octave band filtering is performed on the fly. This allows real-time spatialization of sounds captured by a microphone. The decomposition is performed by a set of second order doubled IIR Butterworth filters [see *Oppenheim and Schafer*, 1975]

Delay line implementation We have seen in Section 3.1.1 that the linearity of DSP operators allows to permute and factorize some operations. In our implementation, all delays have been factorized, so that only two delays are necessary for each path. The two delays correspond to the time of arrival of

[3]The signals are generally anechoic sound files sampled at 44100 Hz.

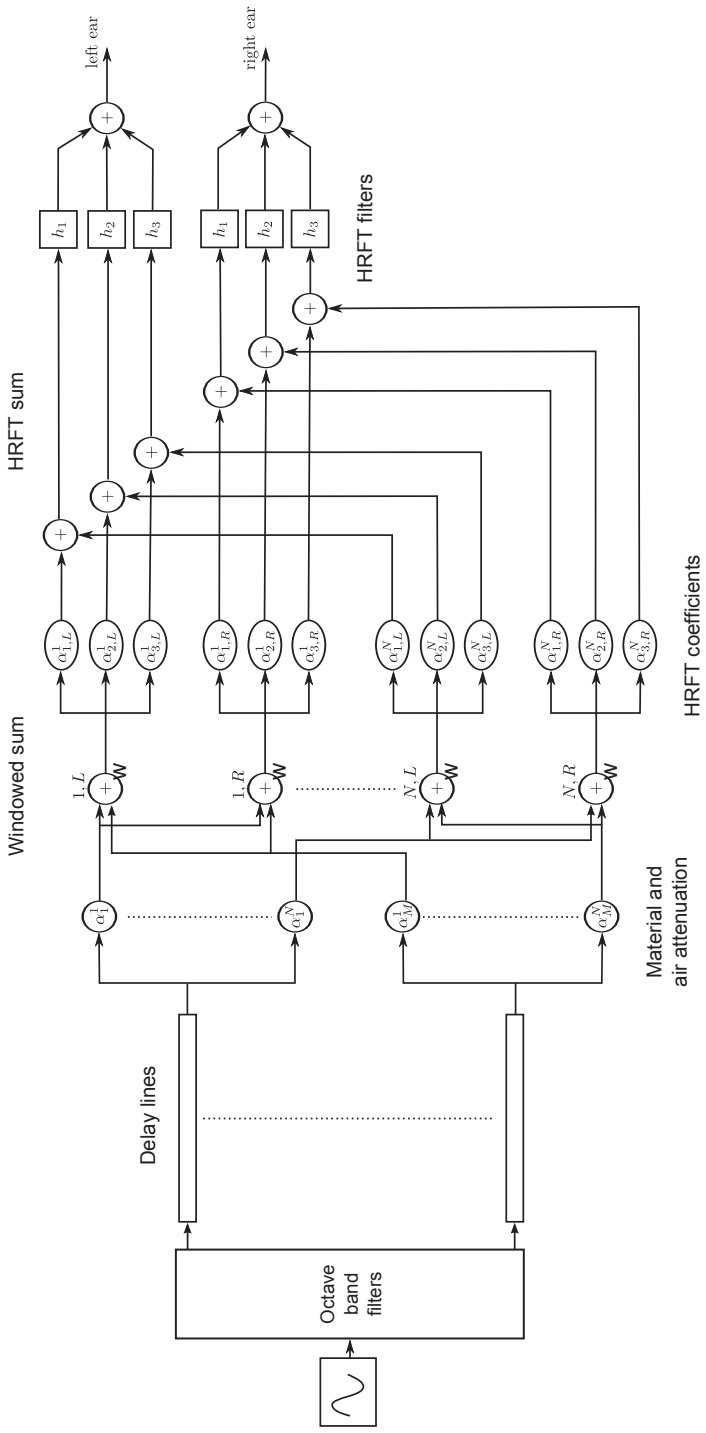


Figure 3.3: Spatialization algorithm including propagation delay, material attenuation, air attenuation and HRTF filtering.

the path at the listener's ears. The total delay of a sound ray τ_{Lefl} reaching the receiver left ear will be:

$$\tau_{Lefl} = \frac{d_R}{c} + ITD_L(\theta_R, \phi_R) \tag{3.7}$$

with d_R the total distance traveled by the ray R in the virtual scene. The total delay of a sound ray τ_{Right} reaching the receiver right ear will be:

$$\tau_{Right} = \frac{d_R}{c} + ITD(\theta_R, \phi_R) \tag{3.8}$$

ITD a function that calculates the Interaural Time Difference (ITD) depending on (θ_R, ϕ_R) the direction of arrival of the ray on the receiver.

The implementation of the delay line presented in Figure 3.3 uses one delay line per frequency band filtered signal. This implementation allows sharing a delay line between many paths. Once the sound is filtered, it is written in a circular buffer whose size must be greater than the maximal distance traveled by a ray in the virtual scene (see Section 1.2.7). Every path of the simulation will have two pointers (one for left and one for right channel) to read the delayed signals.

Two other strategies to build the delays could be implemented. In order to save memory, a single delay line could be used before the octave band filtering process. This would reduce the memory used to store samples, but the filtering of the signal would have to be performed for every path (instead of only one time in our implementation). This strategy would be really computationally consuming. The other strategy would be to have a delay line per path[4]. If the number of paths to reproduce is high compared to the number of octave bands[5], this method would become very memory consuming, and the gain in terms of computational resources would not be very important.

We think that our method is the best compromise between memory occupancy and computational time.

Ray attenuation With the decomposition of sound in octave bands, the frequency dependent attenuation of the ray during its travel is simplified to a multiplication of the band filtered signal by an attenuation coefficient. The coefficient α_i^j represent the values of the filter $h_{\texttt{ES}*}$ presented in Equation 3.6.

Windowed sum during reconstruction

The symbol $\overset{+}{\texttt{w}}$ represents the windowed sum of the signals. It is defined as

$$s_o[n] = \left(\sum_{i=0}^{N_{sig}} s_i[n] \right) \; w[n] \tag{3.9}$$

where s_o is the output signal, s_i, the input signals that are summed, and w represents a windowing function. The usage of the windowed sum will be described in the next section.

[4] This is the strategy used by *Deille et al.* [2006a].
[5] The difference is generally of one or two orders of magnitude.

Spatialization with HRTF The last step of DSP is the binaural spatialization of sounds with the ICA decomposition of the HRTF. The method used in our implementation is identical to the ones described by *Deille et al.* [2006a] and *Emerit et al.* [1995] presented in Section 3.1.4. The filters used in these simulations are order 10 IIR filters. The HRTF basis functions were obtained from ICA applied to a set of 8 individual HRTF.

3.2.2 Spatialization of the most significant paths

The algorithm we have presented is able to process around 150 paths in real-time on current computers[6]. As we have seen in Table 2.1, the number of paths grows exponentially with the complexity of the scene, and the order of reflections. As a consequence, our algorithm cannot be applied to every specular path of a virtual scene. In our implementation, only the most significant paths, from a perceptive point of view, are rendered using the BSA. The selection of the most significant paths is presented in Section 4.3. Here, we present how the most significant paths are processed. One important point for the design of the BSA is that the most important paths in a virtual scene change as the source or the listener moves. This implies that there is not necessarily coherence between the paths spatialized during the simulation.

Transition of the paths

The transition between paths is managed using the windowed sum mechanism defined in the previous section ($\overset{+}{w}$). The audio process of the BSA is based on block processing (see Section 1.2.9), the paths to be rendered cannot be exchanged during a block process; changing occurs necessarily at the beginning of the process of the next block. The spatialization algorithm is assigned a given number of channels depending on the computational resources. This number is static, and defined at the beginning of the simulation. When a new block is processed, four cases may be distinguished for each channel:

- no path is rendered on the current channel, so the path has to be *started* with a smooth transition to avoid artifacts;
- a path was rendered before, but no path is assigned to the current channel for this block, so the previous path has to be *stopped* smoothly;
- a path was previously assigned to the channel, but it is not coherent with the previous path, a smooth transition has to be performed between the two paths;
- the path to be spatialized is coherent with the last processed block, so, no transition has to be performed ($w[n] = 1$).

[6] Around 150 paths can be spatialized while the other algorithms (propagation, perceptive reduction, diffuse field rendering implemented in parallel).

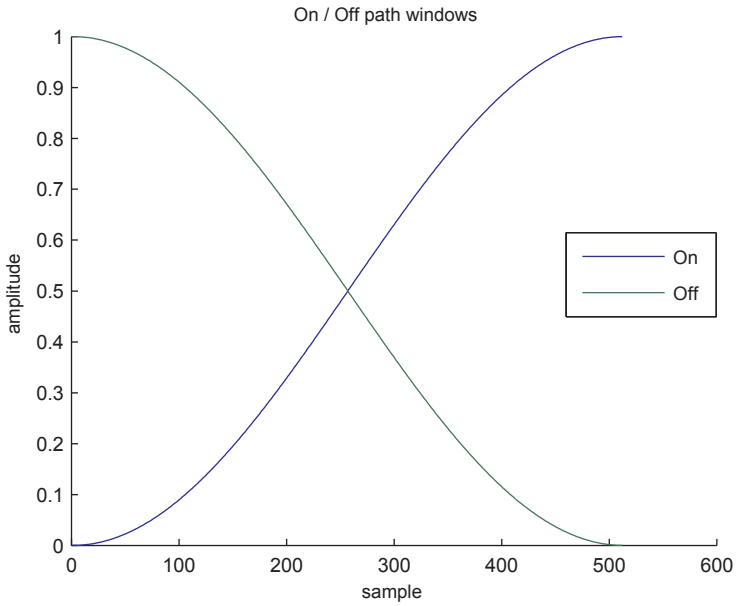


Figure 3.4: Half Hann windows used for starting (blue) and stopping (green) paths.

Figure 3.4 presents the two windowing functions used for *starting* paths and *stopping* paths. The incoherent path processing is simply a double processing of a starting and a stopping path, to create a simple transition between them. When the path is coherent, no window has to be applied.

This interpolation technique has been previously implemented in auralization software, like DIVA (see *e.g.* *Savioja et al.* [1999]).

A fast method to check path coherence

Coherent rays are the rays that have the same history, *i.e.*, that encountered the same sequence of walls during their travel. Some methods have been implemented in the literature in order to check this coherence. One of them, presented by *Stavrakis et al.* [2008], is based on the calculation of the transformation matrix associated with the image source — The translations associated with the displacement of the source are expressed as a matrix transformation. The advantage of this method is that the transformation matrix can be applied to every new position of the source to find the new image sources. This method is useful if the transformation is needed. In our implementation, we just need an information on the history of the rays. So, to check the coherence of the paths, we only need the information about the sequence of walls encountered by the ray. The information of travel of a ray is coded on a 128 bits integer. Every time a ray R intersects a wall, the coherence value R_{CV} is incremented by $(\mathcal{I}_{wall})$, the index of the wall, multiplied by one plus the number of walls in the scene N_{walls} to the power of the order of reflection, o:

$$R_{CV} \leftarrow R_{CV} + \mathcal{I}_{wall}\,(1 + N_{walls})^{o} \tag{3.10}$$

To reduce the size of R_{CV} in Equation 3.10, we give to coplanar walls the same index, as the two image sources issued from two coplanar walls will be the same. This implies that N_{walls} represents the number of non coplanar walls in a scene.

The use of a 128 bits integer restricts the validity of our algorithm when R_{CV} becomes bigger than 2^{128}, that occurs in a very complex scene with 1000 non coplanar walls for reflection orders greater than 13. To reduce the probability of error in the coherence check, we perform a double check:

- First, the order of reflection of the ray R_o is tested, two rays with a different reflection order will not produce the same image source, and thus will not be coherent.

- Second, the check on the coherence value of the ray R_{CV} to discard the paths with the same order of reflection that do not have the same history.

This coherence test is used twice in the global auralization process. It is first used during propagation algorithm with deterministic ray tracing to discard rays with the same history that hit twice the receiver. Then it is used in the BSA to check the coherence of the paths when either the source or the receiver has moved.

Interpolation of the gains

We have seen in Equation 3.6 that the attenuation of a path depends on the medium attenuation (α_m), the distance traveled by the ray (d_R), and the wall absorption coefficients ($\vartheta_{\mathtt{S}}$).

A coherent moving path will share wall absorption as it is linked to the same image source, but the attenuation by the medium and the distance attenuation will be different, that is why the gains of the paths (α_R) are interpolated. Linear interpolation is used in our implementation. Even if the sources are moving fast (see *Maillard* [2009] for implementation of urban noise simulation), the linear interpolation of the gains is sufficient.

When a source or a receiver moves, the direction of arrival of the path on the receiver evolves with time. For this reason, the binaural gains $\alpha_{L,i}$ and $\alpha_{R,i}$ are also interpolated for coherent paths.

Doppler effect using fractional delay lines

The fractional delay lines presented in Section 1.2.8 are implemented in order to interpolate moving paths. The Doppler effect is generally of minor influence in room acoustics, but this method can be used for instance to simulate a fast moving source. For example, a bee flying around the head of a listener is a good example of a Doppler effect.

In our implementation, the interpolation of the delay is linear, which causes no audible artifacts when the source or the receivers moves normally in the virtual scene. For fast moving paths, more complex interpolations have to be implemented [see *Maillard*, 2009]. Generally, the sources and receivers in room acoustics applications have slow motion, so, linear interpolation is sufficient for this kind of application.

3.3 Diffuse field rendering

In Chapter 2, we presented two algorithms to simulate propagation of the sound in a virtual scene. The first one, the deterministic ray tracing algorithm aims at extracting specular contributions. The set of specular contributions contains the most important information from a perceptive point of view for the localization of the sound, and for the characteristics of the reverberation. The auralization of specular paths was presented in the previous section. The other propagation algorithm presented in Section 2.5.2 is the incremental Monte Carlo particle tracing. This algorithm is dedicated to the rendering of diffuse field, *i.e.*, the set of all paths that have at least one diffuse reflection during their travel. We remind that this algorithms produces integrated echograms sampled at 86 Hz. The Room Impulse Response (RIR) is the sum of the specular paths rendered with BSA, and the diffuse field. In order to be consistent with specular response, the diffuse field part of the impulse response has to be re-sampled at the same frequency, *i.e.*, 44100 Hz.

We present here the method used to reproduce a diffuse sound field with windowed white noise from an integrated echogram provided by the MCRT algorithm presented in Section 2.5.2. Then, we present the implementation of a convolution system on GPU for the auralization of diffuse sound field.

3.3.1 Creation of the RIR

We have briefly presented in Section 3.1.5 a method to *fill* an integrated echogram with Poisson-distributed samples [after *Kuttruff*, 1993].

In our approach, the auralization of the diffuse sound field is based on the summation of precomputed white noise windowed by a Hann window.

Smoothing of the echogram with the Hann window

The re-sampling of the echogram from 86 Hz to 44100 Hz with a nearest neighbors strategy would create audible artifacts, due to the discontinuities in the RIR. To perform a smooth interpolation of the bins of the echogram, we chose to sum Hann windows with 50% overlap. Figure 3.5 shows the smoothing of an echogram with weighted Hann windows.

The Hann window is defined as

$$w_h[n] = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N_{hann}}\right)\right) \tag{3.11}$$

with N_{hann} is the number of samples of the window.

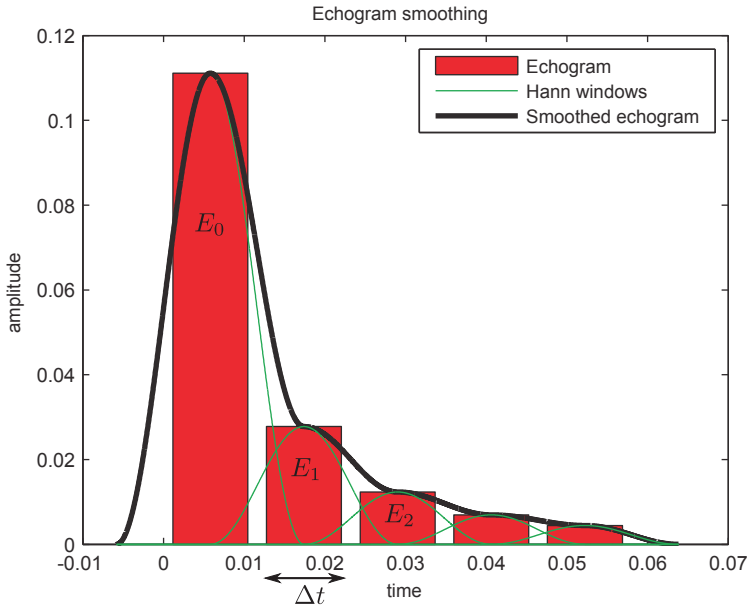


Figure 3.5: Smoothing of the echogram when re-sampling from 86 Hz to 44100 Hz.

Re-sampling of the diffuse echogram with windowed white noise

In order to conserve energy, the impulse response, $p(t)$, created while re-sampling the integrated echogram must satisfy the following equation:

$$\Phi_i = \int_{\Delta t} p^2 dt \tag{3.12}$$

The energy contained in a bin of the echogram, E_i, must be equal to the square of the pressure integrated on Δt. The notations refer to Figure 3.5.

The impulse response is created using pre-calculated noise blocks windowed by a Hann window. For each pre-calculated block of noise, its energy $\Phi_{N,i}$ is calculated as

$$\Phi_{N,i} = \sum_{i=1}^{N_{hann}} \left(n[i]\ w_h[i]\right)^2 \tag{3.13}$$

Where $n[i]$ is a sample generated with normal distribution of mean value $\bar{n} = 0$ and standard deviation $\sigma_n = 1$.

Finally, the resulting IR is constructed as a sum of delayed windowed noise blocks scaled by a factor $\alpha_{n,i} = \sqrt{\Phi_i / \Phi_{N,i}}$. Figure 3.6 presents an integrated echogram and the resulting impulse response obtained by summation of the noise blocks. In our implementation, the non causal samples (*i.e.*, $t < 0$) are discarded. This particular case could be managed using a half Hann window (see Figure 3.4).

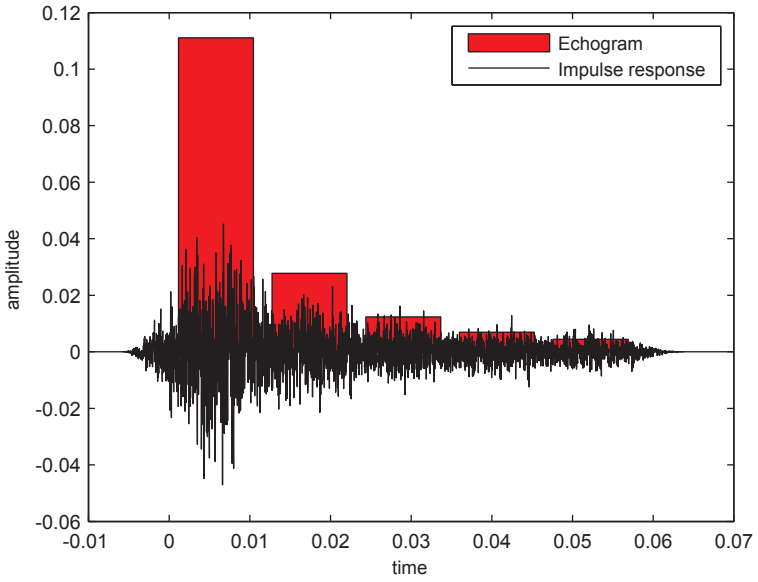


Figure 3.6: Creation of an impulse response sampled at 44100 Hz from a 86 Hz integrated echogram.

Frequency dependent echogram auralization We have seen in Section 2.5.2 that the frequency dependence of the simulation is managed by creating one echogram per octave band. In order to produce frequency dependent RIR, the windowed noise method is applied, but the noise blocks are filtered with octave band filters [7]. The blocks are processed independently for every frequency dependent echogram. Finally, the frequency dependent echograms are summed to recompose the broadband RIR. Figure 3.7 presents the procedure used for the creation of such an echogram. The signal is convolved with six octave band filters. Then, a window is applied with the smoothed echogram for all frequency bands. Finally, the signals are summed to produce frequency dependent RIR.

Binaural rendering A strong correlation of binaural signals is generally localized by the listener inside the head, whereas two uncorrelated signals are perceived as two separate events. In order to provide a binaural auralization, a different impulse response is generated for the left and right channels. In order to create the binaural impulse responses of the diffuse field, the preprocessed noise blocks are thus generated twice — one time for each channel. With this method, the sound field provided to the left and right ears is uncorrelated. According to *Jot* [1992] and *Blauert* [1999], a partially correlated signal could be a good choice to improve the auralization accuracy. Further investigation should be made to find a correlation coefficient that best represents the diffuse field in our model.

[7] The same octave band filters used in Section 3.2.1 for specular paths attenuation.

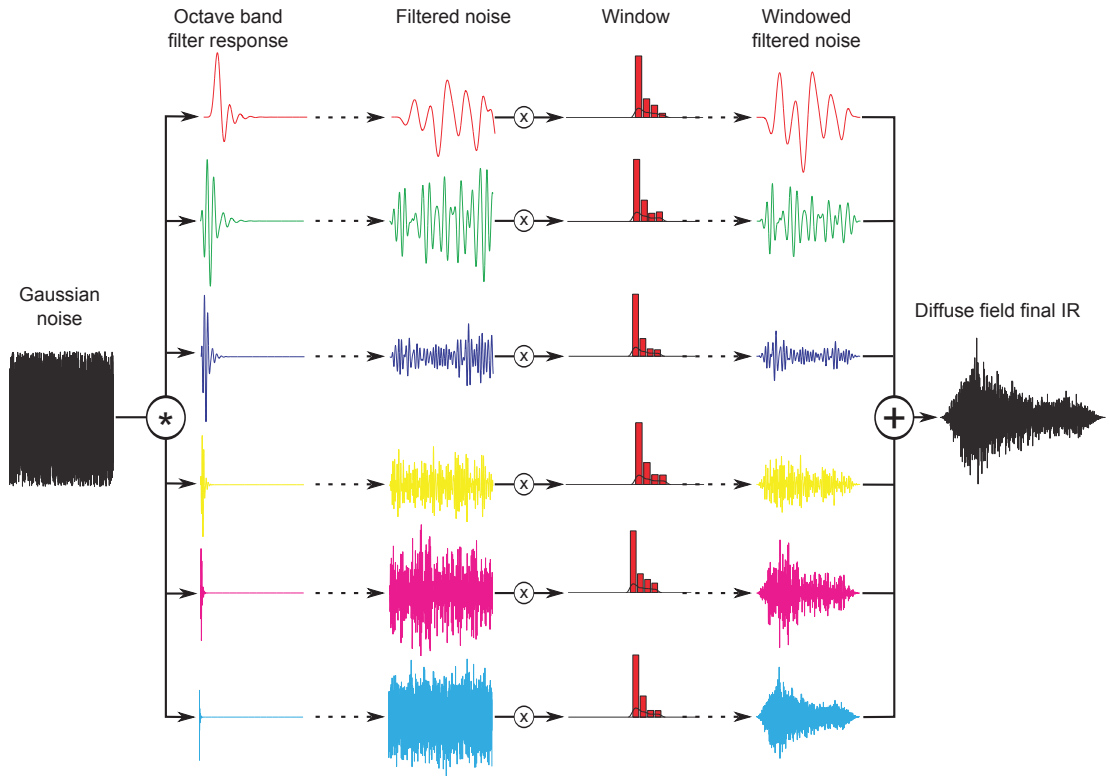


Figure 3.7: Frequency dependent re-sampling of the blocks.

3.3.2 Graphical Processing Unit (GPU) convolution

The auralization of the diffuse sound field is performed by convolution of the anechoic sound by the RIR representing the filter kernel. Even if on recent computer architectures it is possible to perform convolution with long kernels on CPU (see *e.g.* *Siltanen et al.* [2009]), we found that GPU is best suited for this kind of calculation. The convolution algorithm can be implemented as a set of simple parallel operations. For our implementation, we decided to build the development of the convolution algorithm on OpenCL[8] which is a recent standard used to produce reusable source codes on GPU. The implementation of the computation on GPU allows to use all the computation time of the CPU for propagation and specular paths auralization.

Basics of GPU computing

In our implementation, the GPU is a device commanded by the CPU. We have seen earlier that for the processing of 512 samples at a frequency of 44.1 kHz, the processing time of a block must be under 11.6 ms. One of the main bottleneck in real-time GPU algorithms is the rate of data transfer between the CPU and the GPU. Most of GPU applications share the following processing steps: (i) data is first transferred from the CPU to the GPU; (ii) then, processed on GPU; (iii) then transferred back to the CPU. In our application, the data transferred to the GPU is

- the blocks of anechoic signal transferred every 11.6 ms,
- the binaural impulse response, updated upon each modification.[9]

The data returned at the end of the processing is a block of the convolved signal. At this point, it is important to note that the size of the filters, *i.e.*, the RIR, is greater than the size of the processed blocks. Typically, the size of the blocks is around 11.6 ms, while the size of the RIR is 3 s. This means that the result of the convolution of the current block also depends on the convolution of the previous blocks. This is logical, as in real life, the sound that arrives at human ear at a given time is the sum of all delayed sounds that were emitted and propagated in the environment where the listener is placed. Imagine the simple case of an echo in the mountain. A few times after the direct sound is perceived, a second sound is perceived. Numerically, this corresponds to blocks of data that are processed earlier, and stored until they are sent to the listener.

Static convolution procedure

The convolution algorithm on GPU was implemented in two phases. First, the convolution with a static RIR was implemented in order to test the principle of convolution on GPU. This spatialization method is used for the rendering of a static source and receiver in a static virtual environment. This means that the RIR can be pre-calculated, and only the convolution of the anechoic sound with the filter kernel is

[8] OpenCL stands for Open Computing Language. It is a standard supported by Kronos Group that aims at standardizing the routines on various GPU, and more generally, heterogeneous computing platforms, in order to produce reusable source code. The details about OpenCL can be found in the specification document *Munshi* [2009].

[9] Section 5.4 presents the methods used for updating the RIR.

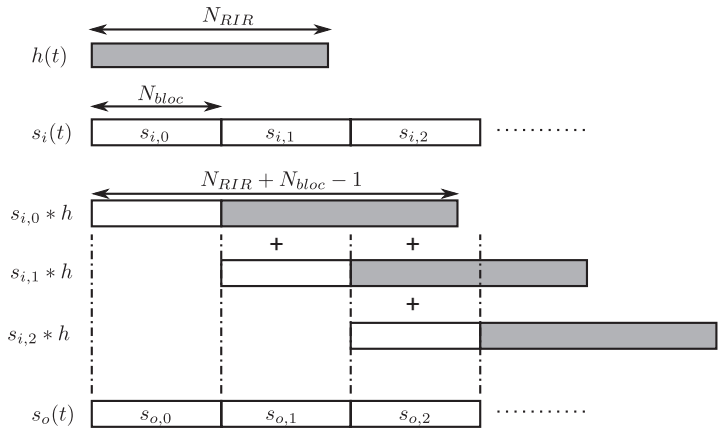


Figure 3.8: Constant-OverLap and Add (COLA) principle.

performed in real-time. We will present in the next section the evolutions of the algorithm for dynamic sound rendering that were implemented in the second phase.

Three main methods may be used for real-time convolution of an audio signal with an impulse response. First, the convolution is performed on filters that have the size of the RIR (usually more than three seconds of the signal sampled at 44.1 kHz). The time domain method could be a good strategy, as one input sample of the original produces one output value. But, this is a really inefficient method to provide the convolution with long FIR filters. The second method is a filtering in the frequency domain. The convolution in the time domain reduces to a multiplication in the frequency domain. As this method is based on Fourier transform, it cannot be processed sample per sample (sample per sample Fourier transform makes no sense).

The third method used to perform the convolution is called COLA[10]. This method is based on the convolution in the frequency domain of short blocks of signal. With this method, it is possible to perform the real-time convolution of the audio blocks provided by the sound card. Figure 3.8 presents the principle of OverLap and Add method. The size of the FFT is defined as $N_{FFT} = N_{RIR} + N_{block} - 1$ in order to prevent temporal aliasing [*Smith*, 1997]. This implies, as shown in Figure 3.8, that the output size of a convolution block is greater than the size of an input block. The mechanism of COLA generates for every block $N_{block} + N_{RIR} - 1$ samples. The first samples correspond to the convolution with the beginning of the impulse response; they are rendered directly. The other samples are summed with the response of the following blocks.

With this method, no windowing has to be applied to the input signal, and the FFT size has to be the next power of two after $N_{block} + N_{RIR} - 1$. It is important to note that in this method, as the response of the filter is static in time, the input blocks do not have to overlap [11]. This method is called by *Smith* [2009] COLA with rectangular window, or COLA with a hop size equal to the block size.

[10] The mathematical definition of COLA can be found in *Smith* [2009].

[11] The name of the algorithm Constant-OverLap and Add (COLA) comes from the fact that the output blocks overlap and are summed (added).

COLA is a more general algorithm that we will present in the next section.

Implementation details We have implemented the previous algorithm in C++ with the GPU computing library OpenCL. To implement the FFT, we used a FFT algorithm based on *Govindaraju et al.* [2008]. The impulse response is calculated and transferred to the GPU, and its FFT is calculated on GPU during the pre-processing stage. In our implementation, we needed to process a RIR of around three seconds. For an impulse response sampled at 44.1 kHz, the nearest power of two value gives $2^{17} = 131.072$ samples. This provides a RIR of 2.97 seconds. To avoid temporal aliasing, the FFT size of the convolution must be $N_{FFT} = N_{RIR} + N_{block} - 1$. So, with an FFT size of 2^{17}, we define the maximal size of the RIR to $N_{RIR} = N_{FFT} - N_{block} + 1 = 130.561$ samples.

The real-time processing of the input blocks is the following:

- the input blocks are transferred from the CPU to the GPU;
- a zero padding is applied to have a signal of N_{FFT} elements;
- the FFT of the signal is performed;
- the RIR and block signal are multiplied in the frequency domain;
- the IFFT is applied to obtain the resulting signal in the time domain;
- the output signal is normalized;
- the output signal is delayed of N_{block} samples and summed with the previously calculated values;
- finally, the first N_{block} values of the signal are transferred back to the CPU.

Some optimizations were implemented to speed up processing. They are presented in Section 3.3.2.

Dynamic convolution procedure

In the context of dynamic rendering, the filter used for the convolution algorithm is no more static. This implies a certain number of changes in the processing presented above; the data transfer and the FFT of the RIR have to be performed in real-time. In our implementation, the frequency of update of the RIR is independent of the frequency of update of the blocks. So, the synchronization mechanism was implemented (see Section 5.3). Then, interpolations are required between two successive impulse responses.

COLA method [*Allen*, 1977; *Allen and Rabiner*, 1977] is used to overlap of half a block. In our implementation, the window used for the input blocks is a Hann window, another window that sums to one can also be used to decompose the input signal in blocks.

For an optimal implementation, we only perform the block processing when the impulse response changes, *i.e.*, when it is updated by the propagation algorithm. COLA algorithm with a 50% overlap is

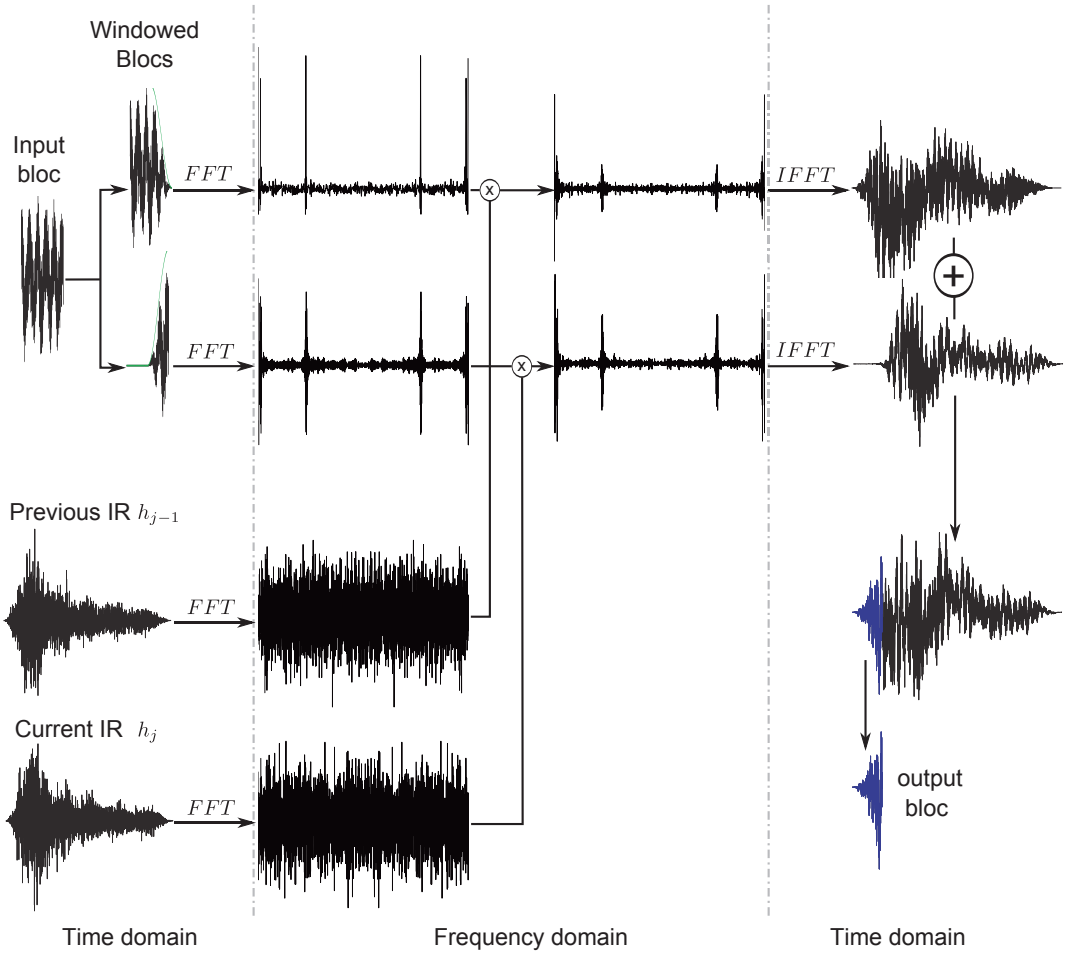


Figure 3.9: Convolution procedure executed on GPU — Only the first part of the convolution is kept as the output block. The remaining samples are kept and summed with the following blocks (see Figure 3.8).

presented in Figure 3.9. Let i be the index of the blocks, and j the index of the RIRs[12]. When the RIR changes, the block is processed twice. The first time, the block processed with a window decaying at the end of the block w_{out}. The signal is convolved in frequency with the previous impulse response h_{j-1}. The result is summed in the accumulation buffer containing the convolutions of the previous blocks. The second time the block is processed, a fade-in window is applied at the end of the block (with 50% overlap). The convolution is performed with the new impulse response h_j. The result is summed in the same accumulation buffer, so that the result of the faded blocks accumulates.

As we want to provide binaural auralization, the COLA algorithm is performed twice, with the RIRs corresponding to the left and right ear.

In the case of fast changing impulse responses, an overlap of a half block may not be sufficient, and audible artifacts might occur. So, we have also extended the method to longer overlap window (from a half block to 2,5 blocks overlap). But, in the case of room acoustics simulations with sound sources moving at normal speed, no difference is audible between the methods. So, the half block method is kept, as it is the least computationally expensive.

We also implemented the Weighted-OverLap-Add (WOLA) method [*Crochiere*, 1980]. In this method, two windows are applied to the signal. The first one is applied to the input signal, before performing the FFT, and the second after the IFFT. The pair of windows is called analysis and synthesis windows. This method is used to compensate the non-linearities present during the filter change. In order to preserve the energy of the signal after such a transform, the windows used are the square root of the windows used for the COLA algorithm, as they are applied twice. In our implementation, the windows are square root Hann windows, also called *MLT sine windows* [*Smith*, 2009]. Informal listening tests have shown that no difference was audible with this method. Consequently, we kept the basic COLA method described above.

Some tests were also performed with static RIR of around six seconds. The static version of our algorithm works well, but the dynamic version reaches the limit of the computation time allowed for a block. This restriction is linked to the hardware used for our tests, and newer generations of GPU would push these limits further.

Optimizations

During GPU development, it was noted that some specific coding approaches that have no influence on CPU may lead to large performance loss on GPU. Some of the best practices in GPU development [*NVidia*, 2009; *Tsuchiyama et al.*, 2010] were implemented in order to speedup the convolution process. Here are some of the optimizations that helped us reach real-time rendering.

Complex to complex Fourier transform This optimization is based on the characteristics of the library used to perform the FFT on the GPU. This library, only performs complex to complex Fourier transform. As the input and output audio signals are real, half of the information transformed is unused.

[12] i and j are different as the frequency of update of the blocks is independent of the frequency of update of the impulse responses.

Based on the two symmetry properties of the Fourier Transform[13], we performed the Fourier transform with two real signals passed as real and imaginary parts of a complex number. Then after the FFT, the transformed signals are decomposed thanks to the symmetrical properties. For the IFFT, the procedure is the same. The details of the complex to complex and invert complex to complex transformations are presented in *Guicquero Le Beyec* [2010].

Computation on local memory One of the main reasons behind GPU code slow execution is due to bad management of the memory accesses. On GPU, the code is executed simultaneously on many processing units that both have shared and local memory. Shared memory is usually slower as it has to manage concurrent access of different processing units. The use of local memory in our algorithms such as complex to complex transform has sped up the process by a factor two.

* * *

The algorithms we have implemented for convolution provide sufficient performances for the spatialization of one source in a virtual environment. This implementation could be improved with an intelligent partitioning of the impulse response and the input signal. These techniques were presented for instance by *Soo and Pang* [1990]; *Gardner* [1995]; *García* [2002]. But, to our knowledge, no implementation of these methods have be made on GPU.

3.4 Combining specular and diffuse field for a unified audio rendering

We have presented in Section 3.3 a method suited for diffuse field auralization, and in Section 3.2 a method for individual specular paths rendering. The problem with the specular rendering algorithm is that it is not able to reproduce every specular contribution provided by the propagation algorithm in real-time. In order to have a complete auralization of the sound field, a choice has to be made on the most significant specular paths to reproduce. And, in order to conserve the energy of the simulation, the paths that are not rendered with the specular algorithm have to be rendered with another method, *i.e.*, it is impossible to discard completely those paths.

One strategy that can be used for the selection of the paths is to follow the well know principle of room acoustics based on early and late reflections (see Section 1.4.1). This implies sorting the rays reaching the receiver by arrival time, and only rendering the first ones with the spatialization algorithm. Another very close approach would be to sort the specular rays reaching the listener by decreasing energy, and again rendering the first rays with the BSA (see Section 3.2).

In our work, we decided to sort the paths based on perceptive parameters. The details of the perceptive sorting of the rays are presented in the following chapter. From a signal processing point of view, our algorithm proceeds as follow:

[13]The imaginary part is antisymmetric and the real part is symmetric

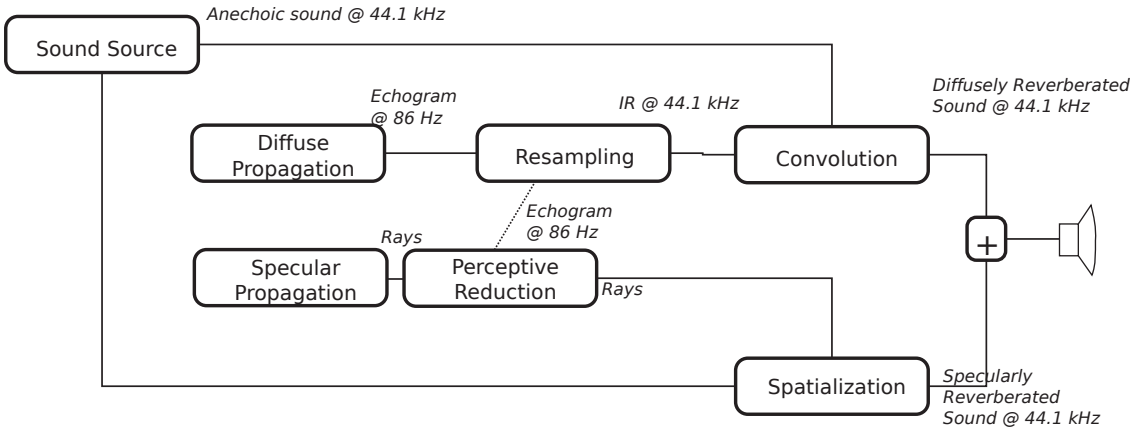


Figure 3.10: Full auralization process including propagation, DSP and perceptive reduction.

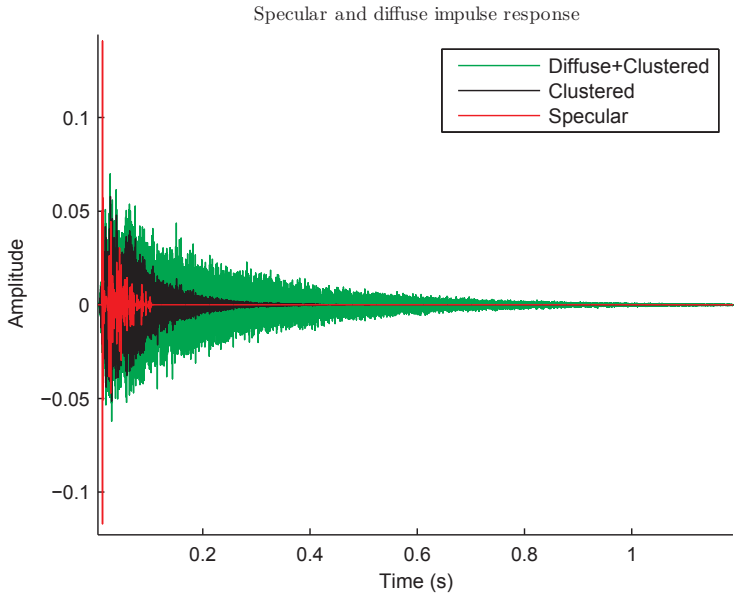
- all specular rays generated by the DRT algorithm (see Section 2.5.1) are sorted according to perceptive parameters;
- the most significant rays are rendered using the BSA;
- all other rays, *i.e.*, the rays clustered by the perceptive algorithm, contribute to the diffuse field, and thus are rendered on GPU with the algorithm presented in Section 3.3.2.

To process the clustered specular rays as part of the diffuse field, their energy is added to the integrated echogram. Figure 3.10 depicts the full auralization procedure with propagation algorithms (sound propagation with specular and diffuse reflexions), DSP algorithms (re-sampling, convolution and spatialization), and the perceptive reduction module. We observe that the spatialization module feeds both the spatialization algorithm with the most significant rays, and the re-sampling module with an integrated echogram containing the energy of the clustered rays.

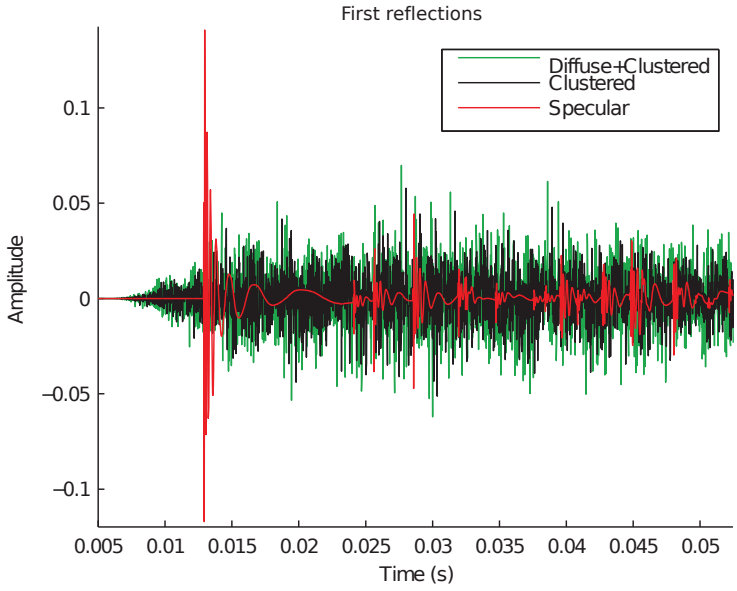
3.5 The Specular/Cluster/Diffuse (SCD) decomposition of the Impulse Response (IR)

Finally, we present the results of our algorithms on the shape and characteristics of Room Impulse Responses RIR. Figure 3.11 depicts the impulse response generated for the simulations of the scene presented in Appendix B.1.3. We presented in Section 1.4.1 the classical way to analyze the IR, with the early, middle and late reflections. Our approach is slightly different in the sense that the most significant reflections rendered by our algorithm no-longer depend on the arrival time. Instead, the choice is based on perceptive parameters. The specular reflections that are not masked[14] by other

[14]Masking schemes for specular reflections will be detailed in Chapter 4.



(a) Impulse response decomposition with SCD.



(b) Zoom on the first reflections of the IR.

Figure 3.11: Two views of the Specular/Cluster/Diffuse (SCD) decomposition of the Room Impulse Response (RIR) generated the scene described in Appendix B.1.3.

reflections are spatialized with the Binaural Spatialization Algorithm (BSA) presented in Section 3.2. This process is heavy, thus, only the most significant paths are rendered using this method.

The other part of the reverberation is rendered using the convolution algorithm on GPU presented in Section 3.3. This part is composed of the sum of the clustered field, *i.e.*, the set of all specular reflections that are masked by the most significant reflections, and the diffuse field, *i.e.*, the set of all acoustic paths that have at least one diffuse reflection.

The advantage of this model is that more computational resources are assigned to the part of the Room Impulse Response (RIR) that has the most significant information from a perceptive point of view. One drawback of the algorithm is that an approximation is made on both the diffuse and the clustered part of the echogram. We can observe in Figure 3.11(a) that the diffuse field is present in the RIR before the first reflection. This is not physically valid, in the sense that no reflection can reach the receiver prior to the direct sound. A way to correct this fact could be to have a thinner decomposition of the diffuse echogram at the beginning of the exchanges, *i.e.*, a non regular sampling of the echogram with sampling period above 86 Hz at the beginning of the echogram.

Informal listening tests have been performed on Room Impulse Response (RIR)s created with our algorithm on the scenes presented in B.1. We found that every time, the direct sound is present, the artifact created by our resampling is not audible.

3.6 Conclusion

Starting from the two propagation algorithms presented in the previous chapter, in this chapter we presented two algorithms dedicated to the rendering of the most significant specular paths and the diffuse field in an auralization application. These rendering methods yields to a new decomposition of the sound field in enclosed spaces. Usually, the room acoustics softwares render the sound field using the separation in early / late reflections. Here, we propose to split the rendering in two parts with first an accurate binaural algorithm for the most significant specular sound paths auralization. Then, a convolution algorithm on GPU for the auralization of the diffuse sound field and the least significant specular sound paths. The details about the implementation of these algorithms and parallel Digital Signal Processing (DSP) structure is presented in Chapter 5. The validation of the implementation of the auralization algorithms is presented in Chapter 6. In the next chapter we will describe the algorithms used to extract these most significant paths from the set of all specular reflections.

4

Perceptive simplifications of pure specular paths

The objective of this chapter is to present a new method to suppress or hide information generated by the propagation algorithms that cannot be perceived by the human ear in order to reduce the computation time needed for the auralization process. We start with a presentation of the various psycho-acoustic effects involved in the localization of a sound in 3D space (see Section 4.1). We also present the works conducted by Hacıhabiboğlu and Murtagh [2008] on the perceptual simplifications for binaural room auralization (see Section 4.1.3). We then present a clustering method for specular paths in room acoustics (see Section 4.3). These algorithms were validated and parametrized using subjective tests (see Section 4.4). Finally, we discuss the benefits of the perceptive simplification method and its integration in our auralization framework (see Section 4.5).

Contents

In order to perform real-time auralization with ray-based algorithms, different solutions can be considered to save CPU time. These solutions can be classified into three categories:

- the type of algorithm used to simulate propagation (ray-tracing, beam-tracing, image-source, ...);
- the digital signal processing methods used for rendering sound;
- the psycho-acoustic methods to avoid the rendering of non-perceivable information.

We study in this chapter the psycho-acoustic simplifications that can be applied to ray-based simulations associated with binaural auralization.

Many studies have been conducted on the masking of sounds, a good summary of these methods is presented by *Blauert* [1999]. More specific studies on the clustering of contributions in room acoustics were conducted by *Bech* [1998]; *Begault et al.* [2001] and more recently by *Hacıhabiboğlu and Murtagh* [2008].

In this chapter, we study the clustering of an echogram generated by ray tracing algorithms. After a state of the art of the different psycho-acoustic mechanism involved in multi-source perception (see Section 4.1), we present the work conducted by *Hacıhabiboğlu and Murtagh* [2008] on the reduction of information for ray-based algorithms (see Section 4.1.3). We then present our clustering algorithm. The clustering is performed in three steps: spatial clustering (see Section 4.2.1), temporal clustering (see Section 4.2.2) and late reflections processing (see Section 4.2.3). We finally present the results of subjective tests (see Section 4.4) that will provide proper parameters for clustering based on the characteristics of the sounds to auralize.

4.1 State of the art

We have seen in Section 2.1.5 that the set of all specular paths can be represented by a set of independent sources called the *image sources*. They represent the reflections of the sound on the walls of the virtual scene and are characterized by their position in 3D space (in Cartesian or polar coordinates), their distance and their orientation. We have seen in Sections 1.3.3 and 3.1.3 that it is possible, with Digital Signal Processing (DSP) operations, to auralize a virtual sound in the space surrounding the listener using Head Related Transfer Functions (HRTF). In this section, we first present the mechanism involved in the localization of a sound source (see Section 4.1.1), then the localization of many sound sources (see Section 4.1.2). Image sources can be seen as a special case of multiple correlated sources.

Not all of the contributions reaching the listener are significant for the perception of the reverberation. In Section 4.1.2, we present a mechanism called *the precedence effect* that defines the masking between sound sources. Finally, in Section 4.1.3, we present a work conducted by *Hacıhabiboğlu and Murtagh* [2008] on the perceptive reduction of information for binaural auralization.

4.1.1 Localization of a sound source

To locate a sound source in 3D space, a listener needs to determine distance and orientation around his head. Polar coordinates are often used in auralization methods as they provide a simple way of expressing some position relative to the listener's head. From a physical point of view, the 3D perception of a sound is possible thanks to the ears, but also to the head and the torso, where complex interactions such as reflection and diffraction occur. These physical considerations lead to the conclusion that the sounds reaching the eardrum of the two ears are different in delay, phase and spectrum, unless located in the vertical plane centered on the head

In virtual reality systems, a special attention should therefore be paid to auralization algorithms, as they must provide coherent phase, spectrum and delay information. Otherwise, the brain of the listener will provide erroneous messages, and the localization will fail.

Distance perception

For a given direction relative to the listener, the perception of the distance is not related to the morphological properties of the listener. As the sound reaches the two ears with a given angle, the perception of the distance is related to the characteristics of the sound. The modifications that occur on a sound wave during the propagation can be modeled as two DSP operations. The first is the attenuation due to the distance of propagation. In terms of DSP operations, it can be modeled by a gain (see Section 2.1.1). The gain corresponds to a level drop of 6 dB as the distance between the source and the receiver is doubled. The other modification of the sound is due to the absorption of the high frequencies by air. This operation can be seen as a low-pass filter depending on the distance of propagation.

These models represent the physics of sound propagation, but psycho-acoustic tests have shown that the perception of the distance is very imprecise in an anechoic room. *Gardner* [1969] showed, for instance, that to perceive the doubling distance of a sound in an anechoic room, a level decrease of 6 dB is not sufficient. Instead, a level decrease of around 20 dB must be applied.

This theory is no-longer valid in room acoustics. As the sound source radiates in all directions, the sound gets reflected on the walls, and reaches the listener many times. In enclosed spaces, the perception of the distance is thus based on the direct to reverberated sound level ratio.

This last observation is important for the design of an auralization application. It implies that direct sound should be rendered with high quality algorithms, respecting the correct delay, phase and spectrum attenuation as it is the most significant element of the auralization.

Localization in the horizontal plane

Much work has been conducted on the localization of sound since 1920. The works conducted by *e.g.* *Stevens and Newman* [1936] showed that the localization error is low for low frequency sounds, then grows up to reach a maximum at around 3000 Hz. The localization error then decreases for frequencies higher than 3000 Hz. For the test procedure, the subjects were blindfolded and pure stationary sounds

Reference	Type of signal	Localization blur (approximate)
Klemm (1920)	Impulses (clicks)	0.75 ° – 2 °
King and Laird (1930)	Impulse (click) train	1.6 °
Stevens and Newman (1936)	Sinusoids	4.4 °
Schmidt *et al.* (1953)	Sinusoids	> 1 °
Sandel *et al.* (1955)	Sinusoids	1.1 ° – 4.0 °
Mills (1958)	Sinusoids	1.0 ° – 3.1 °
Stiller (1960)	Narrow-band noise - $\cos^2$ tone bursts	1.4 ° – 2.8 °
Boerger(1965)	Gaussian tone burst	0.8 ° – 3.3 °
Gardner (1968)	Speech	0.9 °
Perrott (1969)	Tone bursts with differing onset and decay times and frequencies	1.8 ° – 11.8 °
Blauert (1970)	Speech	1.5 °
Haustein and Schurmer (1970)	Broadband noise	3.2 °

Table 4.1: Localization blur for various signals in front of the listener [after *Blauert*, 1999].

were emitted. The error was measured as the distance between the position position pointed by the listener and the real position of the sound.

Another common problem with the localization of sound is the front/rear mis-localization. In the same article, *Stevens and Newman* [1936] showed that above 2000 Hz, the front/rear localization is really improved.

Blauert [1999] presents a good summary of the experiments conducted in the localization of sound. These results are presented in Table 4.1. This table shows that localization does not depend much on the signal emitted. Instead, *Blauert* [1999] presents [after *Preibisch-Effenberger*, 1966] that the localization precision depends much on the position of the source. For a white noise impulse of 100ms, the localization blur angle is 3.6 ° for a sound coming in front of the listener, it falls to 5.5 ° if the sound comes from the back of the listener, and even worst, to 10 ° on the sides (±90 ° on the azimuthal plane).

Localization in the vertical median plane

In the median plane, the Interaural Time Difference (ITD) mechanism does not exist. Thus, the localization is only based on the filtering provided by the Head Related Transfer Functions (HRTF). *Blauert* [1969] showed that the localization of narrow band sounds is not dependent on the position of the sound source. Narrow band signals are located at different positions of the median plane depending on the frequencies of the signal emitted. For instance, sounds with a central frequency of 200 Hz, 2kHz and 16 kHz will be located in front of the receiver, sounds with central frequencies of 1 and 10 kHz will

be located behind the head, and sounds with central frequencies of 500 Hz and 8 kHz will be localized above the head. *Blauert* [1969] showed that there is a correlation between this localization feature and the position dependent attenuation of the HRTF.

4.1.2 Multi-sources localization

In room acoustics, the set of all specular reflections reaching the listener can be seen as different sources at various positions. The positions of the sources are given by the source image algorithm (see Section 2.1.5). All signals reaching the listener's ear have a common origin, the sound emitter. Thus, the signals have a high degree of coherence. All paths reaching the listener carry a delayed and a filtered version of the original sound. In this section we will present the localization of multi-sources that are coherent. Depending on the time between two contributions and the direction of arrival on the listener, three perceptive effects can be observed:

- the contributions are summed — a unique acoustical event is perceived, but its position depends on the position of the two sources;
- one of the contributions masks the second — one of the contributions becomes preponderant, the other one is no-longer perceived by the listener;
- the two contributions are perceived separately — when the delay between two contributions becomes high, they can be perceived as two separate events, this is called an *echo*.

A good review on sound localization in rooms has been provided by *Hartmann* [1983]; *Rakerd and Hartmann* [1985, 1986]; *Hartmann and Rakerd* [1989]. Here are some relevant elements that were collected from these articles.

The effects of room size and absorption

Hartmann [1983] showed that the localization of a sound source in a room is independent from the absorption of the room, but depends on its dimensions. Subjects were placed in the *Espace de Projection* (ESPRO) located at the *Institut de Recherche et de Coordination Acoustique/Musique* (IRCAM) in Paris. ESPRO is a room with variable acoustic. The dimensions of the room can be modified, as well as the absorption of the walls. The subjects were listening to a single 5 0ms pulse, rectangularly gated, of a 500 Hz sine tone. The error of localization was measured for various configurations of the room. The results showed that the localization error for a room with a ceiling at a height of 11.5 m was 3.3 $^\circ$ for the reflecting room (T_{60} = 4 s at 500 Hz), and 3.4 $^\circ$ for the absorbing room (T_{60} = 1 s at 500 Hz). The experiment was repeated with a ceiling at a height of 3.65 m and a T_{60} of 2.8 s. There, the mean localization error falls to 2.8 $^\circ$.

The interpretation of the authors is that the order of arrival of the early reflections has a high influence on the localization of sound. The difference of intensity was measured between the reverberating and the absorbing room. The level of the reflections differs by 7 dB. Thus, the authors conclude that the

time of arrival of the early reflections has a higher influence on the localization of the source than the level difference.

Localization of sounds without attack transients

Other tests were conducted by *Hartmann* [1983] on the localization of sounds without attack. Tests were conducted with continuous sine tone at 500 Hz in the absorbing room presented in the previous paragraph. While the localization was 3.4 ° for an impulsive sound, it reaches 12.6 ° for a continuous sound (with a long onset time of around 7 s). We experimented this effect in our tests (see Section 4.4), where the effects of our clustering algorithm were smaller on sounds with few transients.

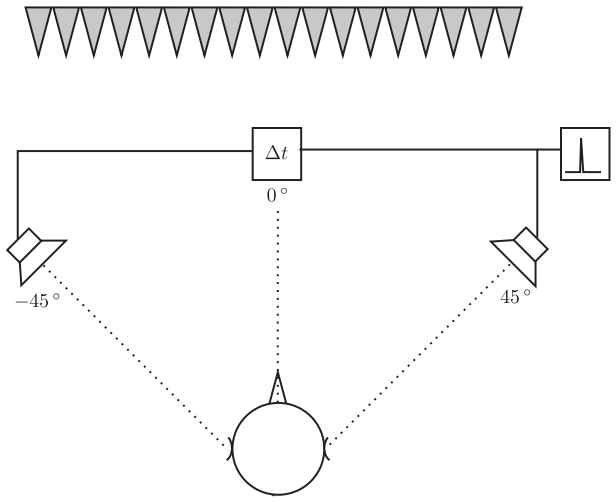
The influence of the onset time of the sound

Further investigations were conducted by *Rakerd and Hartmann* [1985, 1986] on the influence of the onset time of a pure tone sound on its localization. The test above was reproduced with onset durations of 5, 10, 50, 100, 500, 1000 and 5000 ms. The results showed that there is a limit above 100 ms where the onset time no-longer improves the localization precision. This limit is very dependent on the subjects, and experienced listeners tend to have a limit around 50 ms.

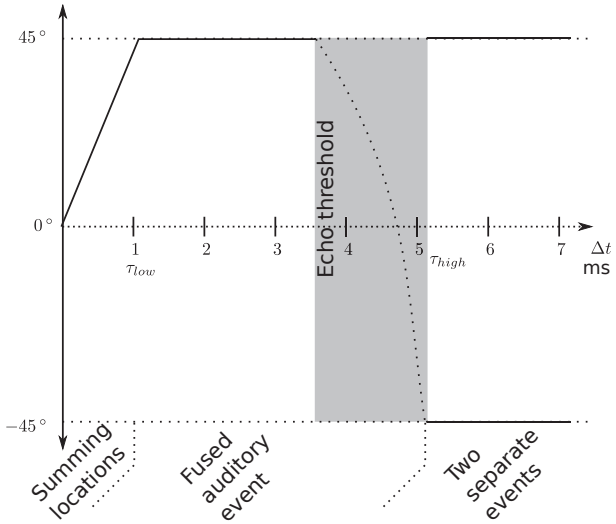
The precedence effect

One of the most studied effects in psycho-acoustics is called the *precedence effect*. The understanding of this effect is important for the comprehension of the rest of this chapter. The name *precedence effect* comes from the original study by *Wallach et al.* [1949]. This section is derived from two reviews about the precedence effect by *Blauert* [1999] and *Litovsky et al.* [1999] that gather and analyze the works on the precedence effect since 1949.

The experimental procedure is depicted in Figure 4.1(a). The listeners are placed in an anechoic room. Two speakers are arranged at equal distance from the listeners, with angle 45 ° and -45 ° . The first speaker emits a signal that represents the direct sound of a room acoustics simulation; this sound is called the lead sound. The second speaker produces a delayed sound (the lag) that represents the first reflection. Figure 4.1(b) represents the ideal perception of the auditory events. When there is no delay between the lead and the lag sounds, the stimuli are perceived as a unique event in front of the listener. As the delay grows to 1ms, the event is perceived as a single event, but the localization moves toward the direction of the leading sound. In the range 1 to 5ms, the events are fused at the position of the leading sound. The lag sound has no influence during this period. After 5ms, the two stimuli are perceived as two separate events. The *echo threshold* represents the delay where the events split from *one fused sound* to *two separate sounds*. As shown in Figure 4.1(b), there is a range around 4-5ms where the localization and discrimination of the lead and lag sounds are hard to distinguish. These values of the precedence effect are measured with impulsive sounds. *Blauert* [1999] showed that with speech signal and musical signals, the *echo threshold* is generally between 30 and 50 ms.



(a) Schematic of the *precedence effect* experiment.



(b) Direction of the events with respect to the lead/lag delays.

Figure 4.1: Classical precedence effect experiment [compiled from *Blauert*, 1999; *Litovsky et al.*, 1999; *Hacıhabiboğlu and Murtagh*, 2006].

This first analysis on the precedence effect shows that some of the contributions provided by the propagation algorithms may not be significant for the final auralization as they will not be perceived by tge human ear. In the following sections we present some masking schemes that used to select the most significant paths.

4.1.3 Perceptual simplification for binaural room auralization

We have presented in Section 2.1.4 various ways to collect rays for ray based propagation algorithms. A different approach to build an optimal collect structure is described by *Hacıhabiboğlu and Murtagh* [2008]. The receiver in this approach is a sphere. The principle of this method is to reorganize the contributions in several clusters in two steps. The first cluster is based on arrival time. Then, for each cluster, the rays are sorted depending on their angles of incidence. The aim of these clusters is to extract the most significant rays from a perceptive point of view, according to the principles of precedence effect.

Early reflections define most of the perceived qualities of a room, even in very reverberant spaces, *i.e.*, with many interfering reflections. To locate a sound, our auditory system gives precedence to the first arriving sound wave — This phenomenon is often called the *law of the first wavefront*.

Precedence effect is only valid during a short time threshold after the leading sound. For broadband signals, such as clicks or white noise burst, the threshold is $\tau_{high} \approx 5$ ms (see Figure 4.1(b)). All sounds reaching the listener before τ_{high} will be perceived as a unique sound. All sounds after τ_{high} will be localized and perceived as a different contribution.

Based on these observations, *Hacıhabiboğlu and Murtagh* [2008] present a two steps perceptual clustering structure, with temporal and spatial clustering functions.

Temporal cluster First, image sources are gathered in a set of clusters $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$. Image sources $\mathcal{E}_i = \{d_{\mathcal{E},i}, \theta_{\mathcal{E},i}, \phi_{\mathcal{E},i}\}$ (in polar coordinates) are gathered in γ_n, if

$$(n-1)\ \tau_{high}\ c < d_{\mathcal{E},i} < n\ \tau_{high}\ c \tag{4.1}$$

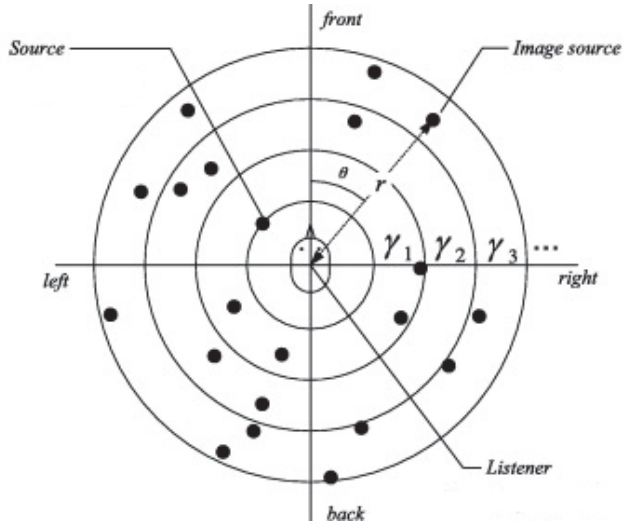
where, c, is the speed of sound in air.

Spatial clustering For each temporal cluster, γ_n, a spatial clustering is then performed, as described in Figure 4.2. This clustering uses an object called *suppressor*. A suppressor is a particular ray that is not masked by any predecessor. The set of all suppressors represents the most significant rays of the simulation. There are never more than three suppressors at a time in the algorithm.

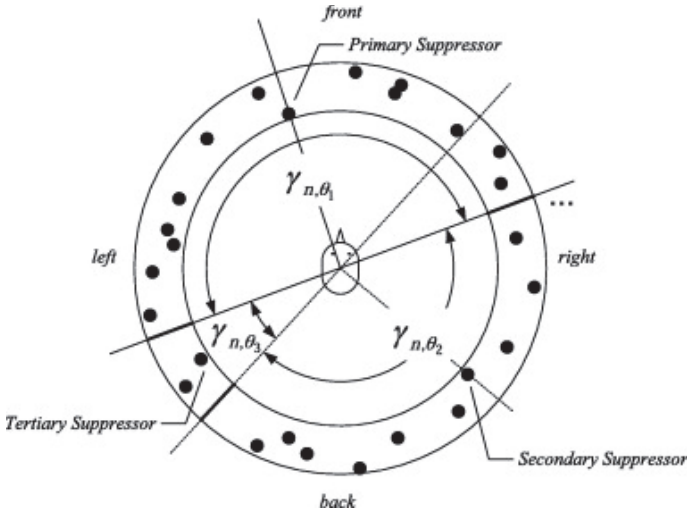
For a given temporal cluster, γ_n, the first image source, $\mathcal{E}'_i = \{d'_{\mathcal{E},i}, \theta'_{\mathcal{E},i}, \phi'_{\mathcal{E},i}\}$, is kept; it is the primary suppressor. All image sources, $\mathcal{E}_j$, that match

$$\theta'_{\mathcal{E},i} - \frac{\pi}{2} < \theta_{\mathcal{E},j} < \theta'_{\mathcal{E},i} + \frac{\pi}{2} \tag{4.2}$$

are gathered in the first cluster $\gamma_{n,\theta 1}$



(a) Temporal clustering of the image sources.



(b) Azimuth clustering of the image sources.

Figure 4.2: The two step clustering algorithm [after *Hacıhabiboğlu and Murtagh*, 2008].

For the remaining sources, the secondary suppressor, $\mathcal{E}_i''$, is found, with $d_{\mathcal{E},i}''$, the minimal distance to the listener. All image sources that match

$$\theta_{\mathcal{E},i}'' - \frac{\pi}{2} < \theta_{\mathcal{E},j} < \theta_{\mathcal{E},i}'' + \frac{\pi}{2} \tag{4.3}$$

are gathered in cluster $\gamma_{n,\theta 2}$.

The remaining sources are gathered in $\gamma_{n,\theta 3}$, the cluster where the third suppressor is found. This two step clustering approach is illustrated in Figure 4.2.

There are two main drawbacks to the method proposed by *Hacıhabiboğlu and Murtagh* [2008]. First, the computation time is too long to be used in a real time algorithm. In the next section, we propose a new implementation of the clustering function combined with a termination criterion in order to reduce the computation time. The second drawback is that the decomposition of the temporal and spatial clustering proposed by *Hacıhabiboğlu and Murtagh* [2008] leads to create many clusters with the same size. During the auralization step, this fixed size structure creates audible artifacts. In Section 4.2 we present a new spatial structure to smooth the masking function and thus creates more irregular clusters. We also propose an incremental parameter to adapt temporally the masking function (the size of the clusters grows each time a new cluster is created). These two improvements lead to suppress the periodic artifacts.

4.2 The masking functions

As shown above, *Hacıhabiboğlu and Murtagh* [2008] have proposed a sequential clustering with a first step of temporal clustering, followed by a spatial clustering (*cf.* Section 4.1.3). Our new clustering algorithm is composed of a single clustering step based on a clustering function, $\mathcal{C}$, and a termination criterion, $\mathcal{T}$. In our approach, the spatial and temporal aspects of the clustering have the same importance; they can be studied independently. The termination criterion is used to set the limit between early and late reflections in the echogram.

Our algorithm is based on the clustering of rays, $R = \{t_R, \theta_R, \phi_R, L_R\}$, with t_R, the arrival time of the ray, θ_R and ϕ_R, the polar coordinates of the arrival direction on the receiver, and L_R, the attenuation level of the ray reaching the receiver. A cluster, γ, is a structure collecting a set of rays sorted by arrival time. A particular ray in a cluster is its first ray; it is called the suppressor, S.

4.2.1 Spatial masking

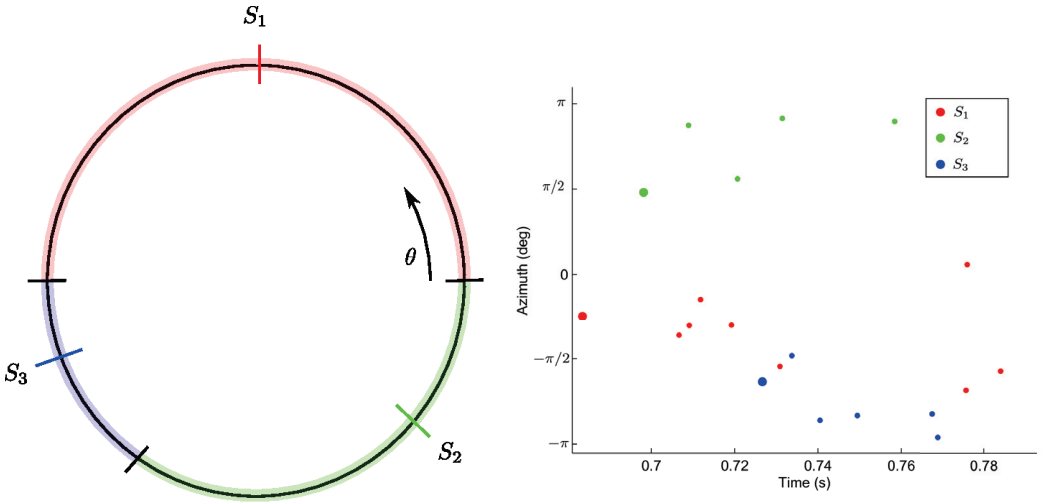
For spatial clustering, we propose a new version of the three suppressor clustering method presented by *Hacıhabiboğlu and Murtagh* [2008]. The first processed ray[1] will be the first suppressor of the algorithm, $S_1 = R_1$. The spatial clustering function $\mathcal{C}_s$ is based solely on the incoming azimuth of the rays following

[1] The direct sound if no occluder is present between the source and the receiver.

the suppressor:

$$\mathcal{C}_s\left(S,R\right) = \begin{cases} 1 & \text{if } \theta_S - \frac{\pi}{2} < \theta_R < \theta_S + \frac{\pi}{2} \\ 0 & \text{else} \end{cases} \tag{4.4}$$

Figure 4.3(a) shows that there will never be more than three suppressors at the same time in our algorithm. An example of spatial clustering is presented in Figure 4.3(b).



(a) Example of suppressor distribution.

(b) Spatial clustering of the rays. The big dots represent the suppressors, the small ones are the clustered rays.

Figure 4.3: Spatial clustering, the suppressors are rays that satisfy both equations 4.4 and 4.5.

4.2.2 Temporal masking

The temporal clustering is based on various studies on the precedence effect [*Litovsky et al.*, 1999; *Blauert*, 1999]. Depending on the stimulus used for the tests, the authors collect a great variety of thresholds to characterize the masking of two sounds. The temporal echo threshold, τ_{low}, is the threshold above which two reflections become audible as a separate auditory event. Above $\tau_{low} \approx 1$ ms, the localisation event depends more on the leading sound than the location of the lagging sound. When the delay is between τ_{low} and $\tau_{high} \approx 5$ ms, the direction of the leading source dominates and the directional discrimination of the lagging source is suppressed [*Hacıhabiboğlu and Murtagh*, 2006].

Begault et al. [2001] have studied the influence of both level ratios, ΔL, and the temporal delay, Δt, between two contributions of an echogram. From the results of their subjective tests, they identified different masking schemes. One is: a single early reflection should be inaudible when its level is less than $\Delta L = 21$ dB below the direct sound at $\Delta t = 3$ ms, and less than $\Delta L = 30$ dB for $13 < \Delta t < 30$ ms. These rules can be seen as masking functions defined by a step function parameterized with Δt and ΔL. To smooth the transition of the masking function, we propose here a new temporal clustering function

$\mathcal{C}_t$ based on a sigmoid function:

$$\mathcal{C}_t\left(S,R\right)=\begin{cases}1 & \text{if } L_R < L_S - \Delta L\left(1/\left(1+e^{-(t_R-t_S-\Delta t)/0.15\Delta t}\right)\right)\\ 0 & \text{else}\end{cases} \tag{4.5}$$

With:

- L_R, L_R the levels of the ray and the suppressor respectively,
- $\Delta t, \Delta L$ the intervals of time and level that parameterize our algorithm.

While performing tests on our clustering algorithm, we have observed that the masking threshold time grows with the time of arrival of the cluster in the echogram. To include this constraint, we defined Δt as a function of the cluster index:

$$\Delta t = \Delta t_0 + i\Delta t_{inc} \tag{4.6}$$

With Δt_0, the temporal parameter of the first cluster and Δt_{inc}, the increment for each cluster.

This method has three advantages:

1. it creates small clusters for the direct sound and first order reflections that are important for the localization;
2. it creates larger clusters where the perceived contributions are less significant;
3. it avoids fixed size clusters that create emergent frequencies during auralization.

Section 4.4 presents the results of our tests for different values of Δt_{inc}. In order to simplify the subjective analysis, the two parameters Δt_0 and ΔL will have fixed values. The parameter Δt_0 is set to 1 ms, that is the lowest time threshold for a fusion of two contributions. We have observed that higher values of Δt_0 can amplify the direct sound so that the perception of the distance between the source and the listener can be distorted. The parameter ΔL is set to 21 dB, according to the observations of *Begault et al.* [2001].

4.2.3 Termination criterion $\mathcal{T}$

The temporal clustering function, $\mathcal{C}_t$, with linearly growing Δt may create gaps at the end of the echogram in the special case of simulations without diffusion. This case occurs when the three suppressors become more significant than the following contributions. To avoid this phenomenon, we stop the clustering algorithm when more than D_{max} rays are present in a cluster. The end of the echogram is thus considered as diffuse field, and processed with the appropriate algorithm.

This criterion has the other advantage to reduce the computational cost of the clustering, as the algorithm does not process systematically all contributions.

4.3 The perceptive clustering algorithm

From equation 4.4 and 4.5, we can define a clustering function depending on time, level and arrival azimuth:

$$\mathcal{C}(S,R) = \mathcal{C}_t(S,R)\mathcal{C}_s(S,R) \tag{4.7}$$

and a termination criterion:

$$\mathcal{T}(\gamma) = \begin{cases} 1 & \text{if size}(\gamma) > D_{max} \\ 0 & \text{else} \end{cases} \tag{4.8}$$

From these functions, we can define a clustering algorithm that takes as input a set of rays R_i, and returns a set of clusters γ_i and possibly an echogram containing late reflections. The first ray of the algorithm becomes the first suppressor $S_1 = R_1$. Then, for each ray R_i, the clustering is tested with suppressors S_1, S_2 and S_3. If a ray is clustered by one of these suppressors S_j, the ray is added to the corresponding cluster, $S_{j\gamma} \leftarrow R_i$. When none of the suppressors clusters the ray, the cluster $S_{1\gamma}$ associated with S_1 is stored. Then, S_2 and S_3 become the first suppressors — $S_1 = S_2, S_2 = S_3$. A new suppressor is created $S_3 = R_i$, and associated with a new cluster $\gamma_j \leftarrow S_3$. Figure 4.4 shows the full clustering algorithm.

4.4 Subjective evaluation

Subjective tests are part of a study presented in one of our publications [*Loyet et al.*, 2009]. This study was performed prior to the creation of the auralization methods presented in the previous chapters. In the following section, we present the auralization framework that was used to perform the subjective tests. In Section 4.5, we discuss the benefits and the parameterization of the perceptive clustering algorithm in our auralization framework. Chapter 6 presents other tests performed on the latest version of the auralization framework. A good perspective to this work could be to perform the same subjective tests on the latest version of the auralization framework, in order to tune finely the parameters of the subjective reduction algorithm.

4.4.1 The binaural auralization framework

Our algorithm was implemented within a binaural auralization framework. The sound propagation is simulated using the Deterministic Ray Tracing (DRT) algorithm described in Section 2.1.5. The binaural auralization is performed with the HRTF filtering method presented by *Emerit et al.* [1995] (see Section 3.1.2). Our clustering algorithm operates on all the rays collected by the receiver. All the generated clusters are sent to the auralization module with the direction of the suppressor associated with the cluster.

The tests were performed on the scene of the third round robin test on room acoustics [*Bork*, 2005a,b] (See Appendix B.1.2). For the ray tracing procedure, the number of rays was set to 10000 and

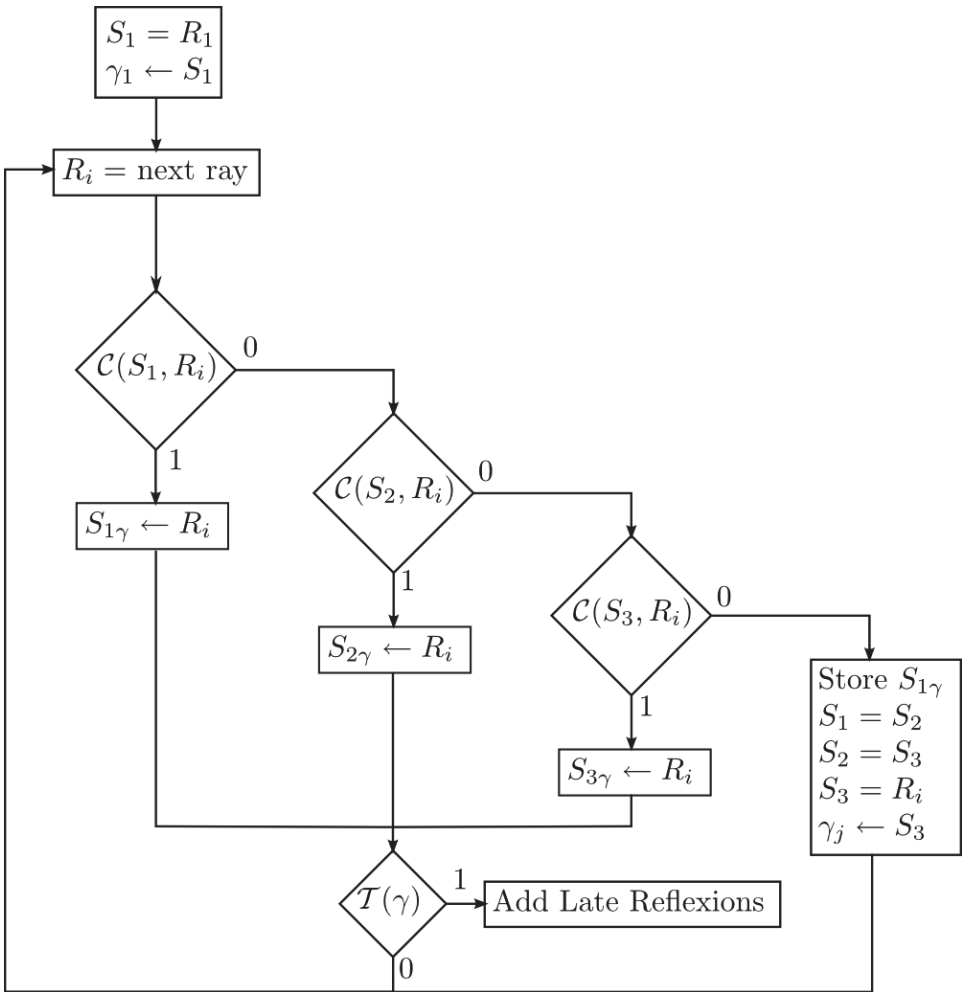


Figure 4.4: Clustering algorithm.

the maximum reflection order to 60. Each time a ray hits the receiver, its travel time, attenuation level, and direction of arrival are stored. A total of 109068 rays were collected and auralized with the binaural module, creating a binaural room impulse response (RIR).

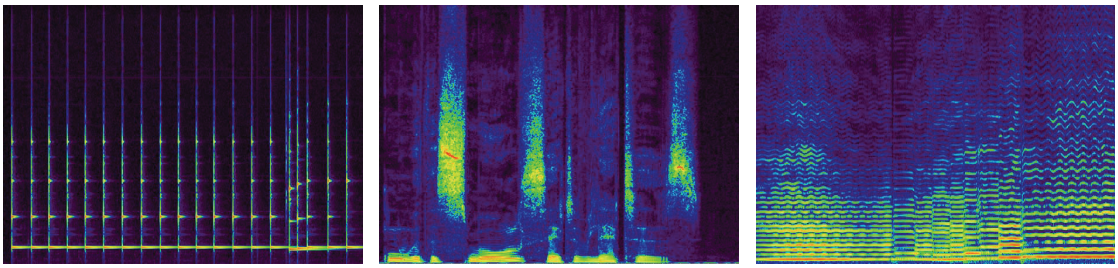
Late reflection processing The termination criterion presented in Section 4.2.3 is used to stop the clustering algorithm. In the implementation of the test procedure, the diffuse field is pre-calculated once for an arbitrary position of the source and the receiver in the room. The end of this echogram is convolved with the anechoic sound to produce the late reflections. This mechanism is useful for propagation systems focused only on specular propagation and auralization.

In the latest implementation of the auralization framework, this mechanism is replaced by the Specular/Cluster/Diffuse (SCD) decomposition (see Section 3.5), *i.e.*, the rays that arrive after the termination criterion are no longer replaced by pre-calculated impulse response. They are summed with the diffuse field to be convolved on GPU.

4.4.2 Population and Test Cases

Eight subjects participated in this experiment. The population was composed of four males and four females between twenty and twenty-five years old. The subjects were members of the staff at CSTB, and half of them were experienced listeners.

The test was composed of three comparative listening experiments. Three sounds with different characteristics were used [*Bang and Olufsen*, 1992]. The first sound is a xylophone solo[2], it contains many transients and is a good test case to hear the characteristics of the room. The second sound is a woman's voice[3], it is a sound that is easier to compare for non musicians. The last one is a cello solo[4], it contains very few transients, and the anechoic sound contains the resonance of the instrument. Figure 4.5 presents the spectrograms of the three anechoic sounds.



(a) Xylophone (b) Voice (c) Cello

Figure 4.5: Spectrograms of the three anechoic test sounds.

A first RIR is generated with the ray-tracing algorithm. It is used for late reverberation factorization

[2]Saber Dance – Kachaturian.
[3]English Female speech.
[4]Variation and Theme No.2 – Weber.

(see section 4.4.1). Then the receiver is moved and a new simulation is run. The binaural RIR is convolved with the three test sounds to produce the reference sounds.

Our clustering algorithm is applied nine times with parameters $\Delta_{t0} = 1$ ms, $\Delta L = 21$ dB, $D_{max} = 300$ rays and $\Delta_{tinc} = \{10^{-4}, 5.10^{-4}, 10^{-3}, 5.10^{-3}, 10^{-2}, 5.10^{-2}, 0.1, 0.5, 1\}$. The values of Δ_{t0} and D_{max} were fixed by previous informal listening tests. ΔL were fixed according to *Begault et al.* [2001] observations on the masking of sounds. The convolution of the nine RIR with the three sounds gives us twenty-seven test sounds.

4.4.3 The test procedure

The test consists in comparative listening tests between the reference sounds and the clustered sounds. The subjects have four levels of classification for a pair of sounds:

1. The sounds are identical: it is impossible to distinguish which one is the reference sound.

2. The sounds are very close: the sounds share the same properties of reverberation time, localization, spaciousness, but have small differences.

3. The sounds are more or less the same: the characteristics of the rooms are the same, but sounds can be clearly distinguished.

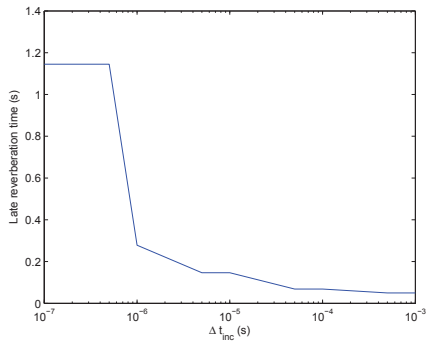
4. The sounds are denatured: the test sound has artifacts or sounds artificial, it cannot be compared to the original sound.

As some of the test sounds have very close properties, the subjects had the possibility to give values of 1.5, 2.5, or 3.5, in order to make a more detailed classification of the sounds. During the evaluation, the test sounds were always played after the reference sound. The subjects were allowed to listen the sounds several times to make their choice.

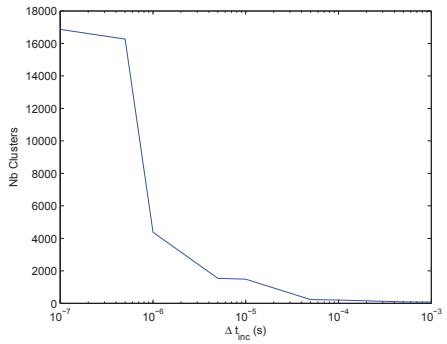
4.4.4 Results

Before the first comparative test, the test consisting in playing twice the same sound was performed. Four of the subjects found that the sounds were identical, the four others said that they were very close. This observation leads to the conclusion that test sounds ranked below or equal to two can be considered as good simulations.

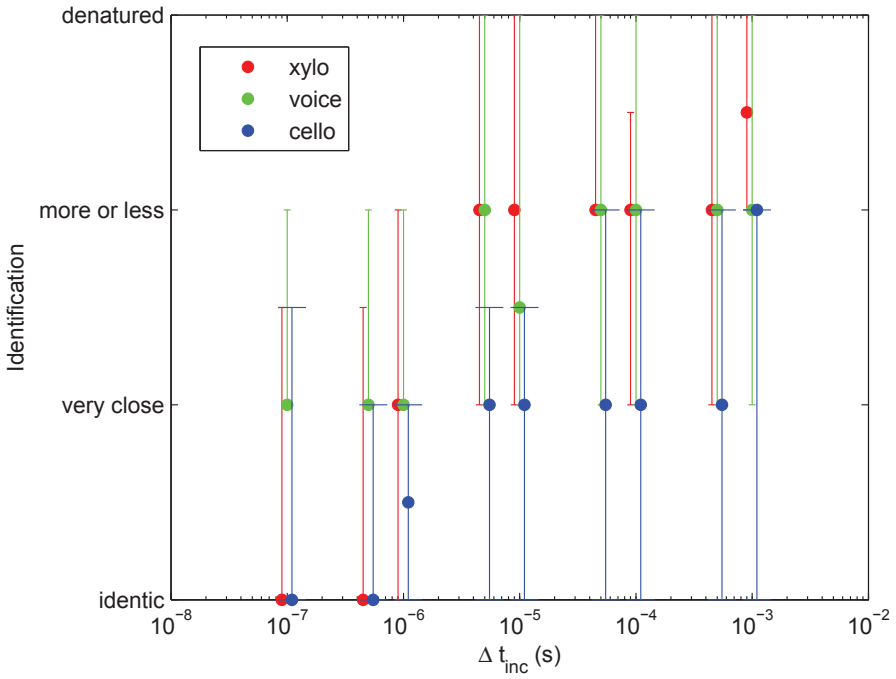
Figure 4.6(c) presents the results of the subjective tests. The dots represent the median value of the tests and the bars the minima and maxima. The first observation is that for $\Delta t_{inc} \leq 10^{-6}$ seconds, our algorithm gives good results on all three test sounds. On the cello, the algorithm can be used for all values of, Δt_{inc}, tested, except 10^{-3}. On the xylophone, *i.e.*, the sound where the influence of the room is highest, our algorithm fails for values of, Δt_{inc}, above 10^{-6} seconds, and creates audible artifacts.



(a) Evolution of late reverberation time.



(b) Evolution of the number of clusters.



(c) Results of subjective tests: Median and min/max interval for different values of Δt_{inc}.

Figure 4.6: Subjective test results.

4.4.5 Discussion

The following remarks were formulated during the test procedure, they can be related to figures 4.6(a) and 4.6(b) that represent the evolution of the late reflection time and the number of clusters as a function of Δt_{inc}.

Nearly all participants pointed that when $\Delta t_{inc} < 10^{-5}$ seconds, the reverberation sounds slightly artificial. It is due to the fact that for these values, the factorization of late reverberation starts at less than 0.2 seconds after the direct sound. This implies that the sound fields of the room cannot be considered as a diffuse field for such values. However, three participants have found that for the voice test, the signal mainly composed of late reverberation produces a more pleasant sound, even if it is not perceived as the original sound.

For the voice, none of the subjects found that the test sounds were identical, and they all found a problem of localization that was not present in the other sounds. This shows that localization is more sensitive for voices than for musical instruments. However, for $\Delta t_{inc} \leq 10^{-6}$ seconds the characteristics of the voice are very similar to the reference sound.

In order to improve this method, further tests should be conducted on the perceptive algorithms.

* * *

We have proposed a new algorithm to cluster information provided by ray-tracing algorithms in order to perform real-time auralization. Subjective tests were performed in order to determine the best values to parameterize our algorithm. We have found that with parameters that fit the three tested sounds, we can reduce the number of auralized contributions by about one order of magnitude, and even further for all sounds which do not exhibit many transients.

4.5 Conclusion and general discussion

At this stage, we have (for real time auralization):

- two propagation algorithms that find all paths (specular and diffuse) between a source and a receiver in a virtual scene;
- a costly binaural auralization algorithm for pure specular paths;
- an efficient convolution algorithm on GPU for the rendering of the diffuse field, and the least significant specular paths;
- an algorithm that selects the most significant specular paths in real-time.

The perceptive reduction algorithm was first developed and tested with a Deterministic Ray Tracing (DRT) algorithm that considers only specular reflections. As we have seen in Section 3.2.2, it is impossible to apply the HRTF filtering to every specular path generated by the propagation algorithms.

This is the reason of our investigations on perceptive simplifications. With this method, the computational resources are concentrated on the most important parts of the signal (from a perceptive point of view). The cost of the perceptive reduction is low compared to the auralization of all specular paths.

The subjective results presented in Section 4.4 are based on a different version of the auralization framework. The diffuse field is estimated once at a given position of the virtual scene, and applied to the end of the echogram. The mixing time, *i.e.*, the limit between early and late reflections is given by the termination criterion, $\mathcal{T}$, presented in Section 4.3. In the current implementation of the auralization framework presented in Section 3.5, the diffuse field is no-longer pre-calculated, nor defined upon the termination criterion. Instead, the diffuse field contains all information generated by the diffuse reflections algorithm (see Section 2.5.2) and all the clustered rays.

In the implementation used for the subjective tests (see Section 4.4), the energy conservation is managed by the suppressors; as a suppressor masks one or more rays, it collects their energy. This leads to an amplification of the contribution at the position of the suppressor. This method was used because there were no overlap in the specular and diffuse fields, and thus the energy of the clustered rays had to be present in the resulting echogram.

With the current implementation, a different strategy is applied. As diffuse and specular fields overlap, the clustered rays no-longer contribute to the energy of the suppressor. Instead, they are added to the diffuse echograms, and processed on GPU, as presented in Section 3.4.

Informal listening tests were performed with this method, but further perceptive listening tests should be made to validate this approach and to find the appropriate clustering parameters for the auralization with diffuse field.

A last parameter was omitted in the presentation of the perceptive clustering algorithm: the frequency. As the propagation is performed by octave bands, the information given to the clustering algorithm has the same format. The temporal masking function, $\mathcal{C}_t$, (*cf.* Equation 4.5) is based on the level difference between a ray, R, and a suppressor S. Three strategies were tested to determine the level of the rays:

- the energy of the ray is the energy of an arbitrary frequency band — this method is not satisfactory because some important information may be lost if the signal is filtered in the given band;
- the energy of the ray is the sum of all frequency bands — this represents the global energy of a ray;
- the energy of the ray is the maximum of the energy of the bands — this method allows the masking of contributions that have heterogeneous frequency content.

The last two methods were tested, they provide different clusters for a given set of specular reflections, but, from a perceptive point of view, the differences were not noticeable during our tests.

The next chapter presents the details of implementation of the different modules that compose an auralization framework. It also presents their interactions, and the global software structure.

5

Efficient task scheduling

In this chapter, we present the technical aspects related to our auralization framework. After a short review of some real-time auralization methods (see Section 5.1), we present the structure we developed for real-time Digital Signal Processing (DSP) operations. This structure called audio graph *aims at executing the DSP operations in parallel on recent computers (see Section 5.2). We then present the global computational structure of our auralization framework (see Section 5.3). After the observations of the previous chapters, we found that all the elements necessary for auralization can be seen as independent modules. We present those different modules, their interactions and the scheduling structure that pilots themselves. Finally, in Section 5.4, we present a new method for interactive progressive update of the audio rendering. This method updates the most significant parts of the Room Impulse Response (RIR) while the listener is moving in the virtual scene. When he stops moving, the remaining parts of the RIR are progressively updated.*

Contents

Gordon Moore explained in 1965 that the complexity of semiconductors doubled every eighteen months at a constant cost since 1959, the date they were invented. This exponential augmentation has shortly after been called the *law of Moore*. This proved to be true until 2004 when the evolution lost speed, mainly due to the effects of thermal dissipation. As a consequence, the structure that has been adopted by most of the constructors is the multiplication of calculation cores. Nowadays, the tendency is the octo-core. Moreover, there is an increase in the use of other calculation units such as Graphical Processing Unit (GPU).

These innovations make the use of software based on sequential operations only obsolete. This is why an important part of the conception of our auralization framework is based on parallel processing of the algorithms. Section 5.2.2 presents the parallelization of DSP operations on both CPU and GPU. Section 5.3.2 presents the global parallel structure of the application, and how the execution of the modules is scheduled.

Our application was developed in C/C++ [*Stroustrup*, 1997]. The main external libraries used for the development were PortAudio[1] for the management of the sound card, OpenCL [*Munshi*, 2009] for the computation on GPU, QT[2] for the Graphical User Interface (GUI) and OSG[3] for the visualization of the scene. The developments were conducted on a Microsoft Windows platform using the Visual Studio 32 bits compiler.

The computer used for the development and the tests is an Intel Core 2 Quad at 2.66GHz with 4 Gb of Ram and two NVidia 8800 GTX GPUs.

5.1 State of the art

Many articles have been published in the field of real-time auralization, including (among others) *Svensson* [2002]; *Funkhouser et al.* [2004]; *Schwark et al.* [2004]; *Lesoinne et al.* [2006]; *Deille et al.* [2006a]; *Lentz et al.* [2007]; *Kajastila et al.* [2007]; *Röber et al.* [2007]; *Lauterbach et al.* [2007]; *Kapralos et al.* [2007]; *Noisternig et al.* [2008]; *Siltanen et al.* [2009].

We briefly present in this section two of them; the first [*Deille et al.*, 2006a] presents a real-time auralization method with pre-calculation of the propagation. The DSP methods presented in this article greatly inspired us for the development of our auralization system (see Section 3.1.3 and 3.2). The second article [*Siltanen et al.*, 2009] presents an original propagation method in the frequency domain and an implementation on recent GPU architectures.

[1] http://www.portaudio.com/
[2] http://qt.nokia.org/
[3] http://www.openscenegraph.org/

5.1.1 Real-Time Acoustic Rendering of Complex Environments Including Diffraction and Curved Surfaces [after *Deille et al.*, 2006a]

Propagation method

This article presents an auralization framework including the effects of diffraction and curved surfaces. The propagation is performed using an adaptive beam tracing algorithm [*Funkhouser et al.*, 2004]. The beam tracing algorithm is a geometric propagation algorithm of the same family as the Deterministic Ray Tracing (DRT) algorithm (see Section 2.1.5). The major problem with DRT is that the specular contributions found are linked to the corresponding image sources. Image sources are relative to planar surfaces, thus, DRT is not suited for environments with curved surfaces.

Diffraction is processed up to the first order of diffraction. In a first pass, diffracting edges are found using beam tracing. Each diffracting edge then becomes a new source, the cone of diffraction [see *Keller*, 1961] is then discretized in new beams and the propagation continues.

The late reverberation is calculated with an artificial reverberation algorithm using the mean free path, and the reverberation time as input parameters.

Real-time audio rendering

The audio rendering algorithm is composed of two steps. The first operates on specular and diffracted paths. The second is an artificial reverberation algorithm based on the mean free path and reverberation time calculated during the propagation step.

The binaural auralization algorithm was presented in Section 3.1.2. It gave us a good basis for the implementation of our Binaural Spatialization Algorithm (BSA). However, as the spatialization algorithm is a computationally expensive process, a choice is made on the paths to render. In this article, the choice is based on the time of arrival of the paths. The maximal number of paths to reproduce is estimated on user's computer, and, during runtime, the paths are sorted, and rendered according to their time of arrival. The advantage of this sorting method is that all the paths that are rendered using the spatialization algorithm are located at the beginning of the impulse response. Thus, the auralization of the diffuse field is parametrized by the time of the last specular path rendered. The drawback is that the auralization of specular paths is not based on perceptive parameters. Thus, a part of the expensive binaural spatialization process is applied to paths that are not significantly perceptual, whereas other significant specular paths are rendered statistically with the artificial reverberators.

Precomputed data

The main drawback of this method is that the propagation step is pre-calculated. In order to reach real-time requirements, the authors focused the real-time rendering on the auralization algorithm. The propagation is performed for a static position of the source, and a dynamic version of the receiver. To have a moving receiver, the propagation paths are calculated and stored on the nodes of a grid. During the real-time auralization, the position of the listener on the grid is fixed, and the auralization is based

on the interpolation between the nearest points to the listeners.

5.1.2 Frequency domain acoustic radiance transfer for real-time auralization [after *Siltanen et al.*, 2009]

Recently *Siltanen et al.* [2009] proposed a new radiance method using the RARE (see Section 1.1.5) and based on frequency domain propagation. This method is implemented on recent GPU architectures. We present in this section the choices that were made by the authors on the distribution of the calculation modules between CPU and GPU.

Propagation method

The methods used for propagation is called the *acoustic radiance transfer method*. It was introduced by the same authors in a previous article [*Siltanen et al.*, 2007]. The acoustic radiance method starts from the RARE presented in Section 1.1.5. The principle is the following:

- the geometry of the scene is divided into patches;
- acoustic energy is propagated from the (static) sound source in the direction of all un-occluded patches;
- the patch with the highest undistributed energy radiates as if it was a sound source, and is marked *distributed*;
- the energy is weighted by the form factor (see Section 2.1.6) between the patches and the BRDF of the emitting patch (see Section 2.1.3);
- the propagation step is repeated until the highest undistributed energy falls under a very small threshold.

With this method, the walls of the scene store the time-dependent energy distributed by the sound source. The last phase of the algorithm is a collect phase; each time the listener moves, a collect of the energy on every un-occluded patch is performed.

The authors proposed in the original approach to perform the propagation in the frequency domain. As all operations applied to sound propagation can be seen as linear operators, the propagation operations are replaced in this case by their Fourier transform (see Section 1.2.2).

Pre-computation

During the pre-computation stage, all subdivision and propagation steps from the source are performed in the frequency domain. This implies that the sound source is static. Apart from the static source constraint, the main drawback of this method is its pre-calculation time, it goes from around fifteen minutes for a simple cube scene (twelve patches after subdivision) to 99 hours and 12 minutes for a complex concert hall (1176 patches).

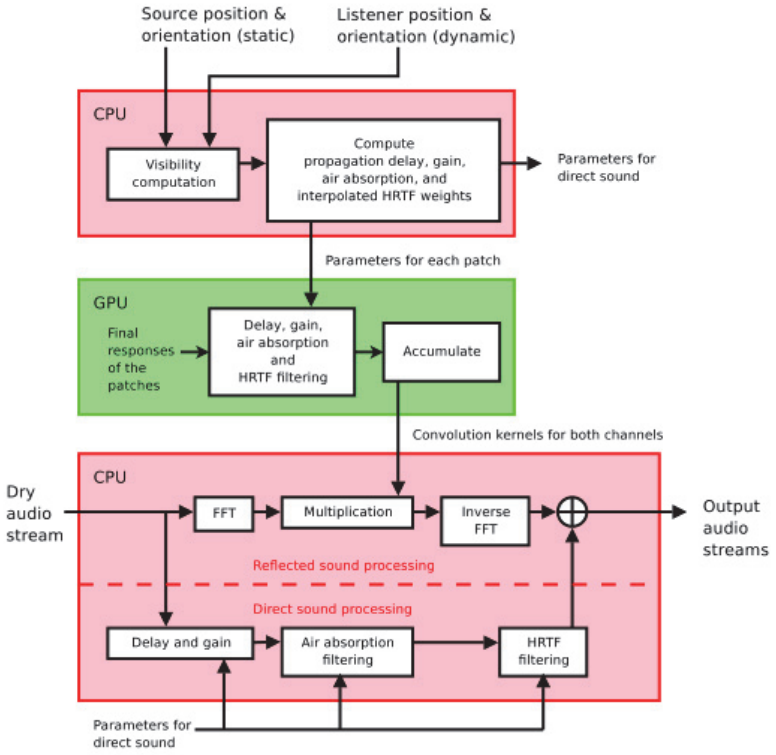


Figure 5.1: Acoustic radiance transfer method for real-time auralization computation on GPU and CPU [after *Siltanen et al.*, 2009].

Run-time computation

One of the strengths of the method is the implementation of a part of the real-time process on the GPU. Figure 5.1 presents the distribution of the calculations between the CPU and the GPU. As the listener is moving in the scene, the visibility computation is performed on the CPU. During this step, the parameters for each patch are gathered on the CPU and transferred to the GPU. DSP operations in the frequency domain are performed on the GPU, *i.e.*, delays, gains, air absorption and HRTF filtering. The results are accumulated to form the frequency response of the room. This response is transferred back to the CPU to be convolved (multiplied in frequency domain) with the anechoic audio stream. The processing of the direct sound is also performed on the CPU in order to have a more accurate auralization (see remarks in Section 4.1.1).

This approach uses the GPU to compute most of the auralization elements, including the specular part.[4] The convolution with the resulting frequency response is then performed on the CPU.

[4]Note that the acoustic radiance transfer method avoids the use of perfect specular paths. Instead, specular paths are represented as beams around the specular direction.

In our approach, we have the opposite strategy, the rendering of the pure specular paths is performed on the CPU, while the convolution with the diffuse RIR is performed on the GPU as presented in Section 3.5. In the next sections, we present the implementation details of our algorithms running both on CPU and GPU.

5.2 Digital Signal Processing (DSP) audio graph

From a computational point of view, DSP can be represented in the form of a graph. This representation is for instance used in commercial software such as Matlab Simulink [5]. The aim of this representation is to hide the computational part of every DSP that composes a system. The operations can be seen as black boxes with a given number of inputs and outputs. A DSP system is thus defined by a set of interconnected nodes, each node implementing specific operations. As all the signal processing operations can be represented by a graph, and since we deal with audio signals, we will refer to the overall signal processing algorithm as the *audio graph*. We present in this section the basic concepts contained in the audio graph (see Section 5.2.1), and the details about the parallel execution of the nodes (see Section 5.2.2). A complete description of *audio graph*, its implementation details, and some use cases can be found in *Loyet* [2007].

5.2.1 Presentation of the audio graph

The two main entities that compose the audio graph are the *audio graph* itself, and the *audio nodes*. We present how they can be used to build an *audio application*, *i.e.*, a DSP system that generates or processes audio information.

Audio nodes

An audio node refers to the computational representation of a DSP operation. It can be either a simple operation such as a pure tone sine generator, a wave file reader, a noise generator, a delay, a mixer, a filter ... But it can also be a complex process such as a binaural auralization process or a GPU convolution. The aim of the audio node structure is to hide the complexity of the process, in order to focus on the design of the application. The design of the application is performed by interconnecting audio nodes with a given number of buffers[6]. An audio node is composed of:

- a process function that defines the DSP operation of the node,
- a given number of input ports,
- a given number of output ports.

Note that the number of ports may be zero, *e.g.* in the case of a signal generator, there is no input port. In the case of a node that writes a signal to a file, there is no output port.

[5] http://www.mathworks.com/products/simulink/
[6] A buffer is a shared part of memory between two audio nodes

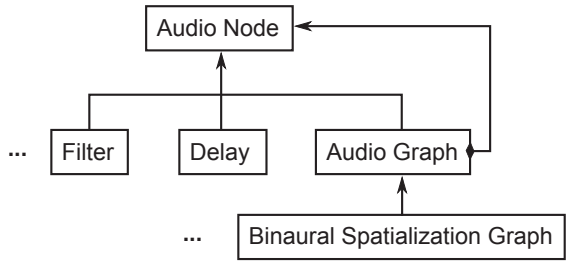


Figure 5.2: UML representation of the structure of an audio node, an audio graph, and their inheritances.

The ports are the elements of the node that allow connections between nodes. Depending on the arrangement of the nodes, the processing can be either sequential or parallel (see Section 5.2.2 for details about the parallel execution of nodes).

The main characteristic of connection ports is that they can be added or removed dynamically, thus providing dynamic modular processing. Take the example of a port associated with each specular path to auralize. When the path disappears, the port is removed, when a new path is found, a new port is created. The following constraint is applied while connecting ports: an output port can be connected to one or more input ports. But, an input port can receive the information provided by only one output port. Therefore, all input ports connected to a given output port can read (and read only) the signal resulting from the current processing node.

This constraint thus implies a direction in the processing of the nodes. The Binaural Spatialization Algorithm (BSA) presented in Figure 3.3 shows an example of arrangement of interconnected nodes, the nodes are processed from left to right.

Audio graph

An audio graph is a computational structure that contains interconnected nodes. This structure manages the connection between nodes, and thus the way the nodes are executed. But, in order to have a totally modular structure, an audio graph is also an audio node. This trick of conception is called a *composite object* in oriented object programing [*Gamma et al.*, 1995]. Figure 5.2 shows the Unified Modeling Language (UML) representation of the structure of node and graph. It shows that a graph is both a child (in the sense of object oriented inheritance) and a container of nodes. The aim of this arrangement is to provide a hierarchical structure. Note that a graph may also contain other graphs, without any restriction of levels of inclusion.

As a graph is a node, it also contains input and output ports. The nodes belonging to a graph may connect to these ports, in order to have interactions with the nodes located outside the graph.

Audio application

An audio application is a special audio graph. It represents the main graph of a DSP system, the graph that contains all other graphs and nodes. It is a special graph as it must also interact with the sound

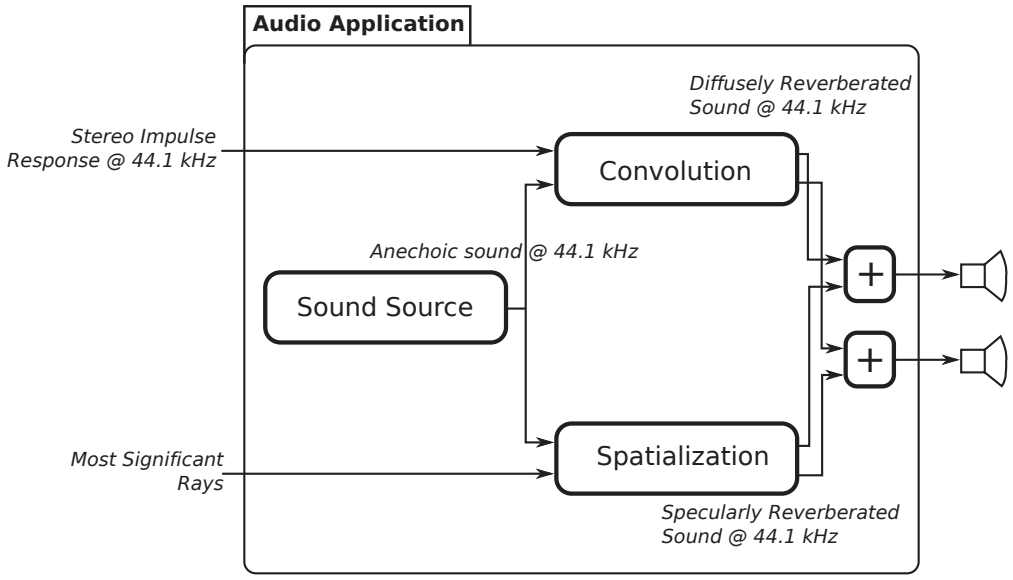


Figure 5.3: The macroscopic view of the audio application.

card.[7] The audio application is for instance the graph that starts/stops the audio stream, that receives input signals provided by microphones, and that sends the processed signal back to the sound card.

With this hierarchical representation of an audio graph, it is possible to present different levels of analysis of the auralization process. Figure 5.3 presents a macroscopic view of the audio application. It hides the whole complexity of the algorithms, and gives a quick view of the main modules implemented.

A detailed view of the Binaural Spatialization Algorithm (BSA) is presented in Section 3.2 (see Figure 3.3). It may be seen as an audio graph (and is implemented as such). The GPU convolution algorithm may also be seen as an audio graph. Details of implementation are presented in Section 5.2.2.

5.2.2 Parallel execution of the nodes

The general parallel structure

The parallel processing of audio nodes, as most of parallel processing, implies the use of lightweight processes called threads. While a sequential execution is executed on only one processor, a parallel code spreads the execution kernels on all the cores of the CPU. In the case of a graph processing, the nature of the graph shows if it is possible to execute tasks in parallel. If two nodes share the same predecessor, then they can be executed in parallel. This is one of the advantages of the graphical representation of DSP operations. The parts that can be processed in parallel can be seen directly from the graphical representation — this is more difficult when presented in the form of equations.

However, the central problem of parallel execution is the concurrent access to data when different

[7] Through PortAudio Application Programming Interface (API).

cores process the same audio buffer. The solution to this problem is to schedule the calculation so that there are never two simultaneous processes that write to the same buffer. The graph structure presented in the previous section solves this problem. Indeed, each node of a graph is an independent element, it only needs to have a reading access to his predecessor, and a writing access to its current buffers.

The thread structure used to execute the parallel processing of our audio graphs is called *active objects* [*Foote and Yoder*, 1998]. These are threads attached to objects[8] that have two states: *waiting* or *processing*. This method is interesting because it is very close to the implementation of the audio nodes. When the other nodes are processing, the current node is set in *waiting* state. When all the predecessors of the node have finished their process, it changes to the *processing* state.

Control of parallel execution

The first implementation of the audio graph presented in *Loyet* [2007] used one active object per audio node. This method works, but, as the number of nodes in an application becomes high, a great majority of the threads are in a *waiting* state. This could be memory consuming. A good design strategy for a multi-thread application is to have a number of threads equivalent to the number of cores of the computer where the application is executed. To solve this optimization problem, we implemented a module that contains a number of threads (this number is defined at startup). This module contains a list of all the nodes that are waiting for processing, *i.e.*, the nodes whose predecessors have finished their process. Every time the process of a node terminates, this list is updated, and the thread assigned to the node that has just finished is re-assigned. This method is an efficient way to parallelize a complex graph with a fixed number of threads.

GPU calculation nodes

Let us remind from Section 3.3.2 the main calculation steps for the convolution on GPU — for the process of one audio block of size N_{block}:

- the input block is transferred from the CPU to the GPU,
- the RIR is transferred from the CPU to the GPU,
- a zero padding is applied to yield signals of N_{FFT} elements,
- the FFT of the block and the RIR are performed,
- the RIR and block signals are multiplied in the frequency domain,
- the IFFT is applied to have the resulting signal in the time domain,
- the output signal is normalized,
- the output signal is delayed of N_{block} samples and summed with the previously calculated values,
- finally, the first N_{block} values of the signal are transferred back to the CPU.

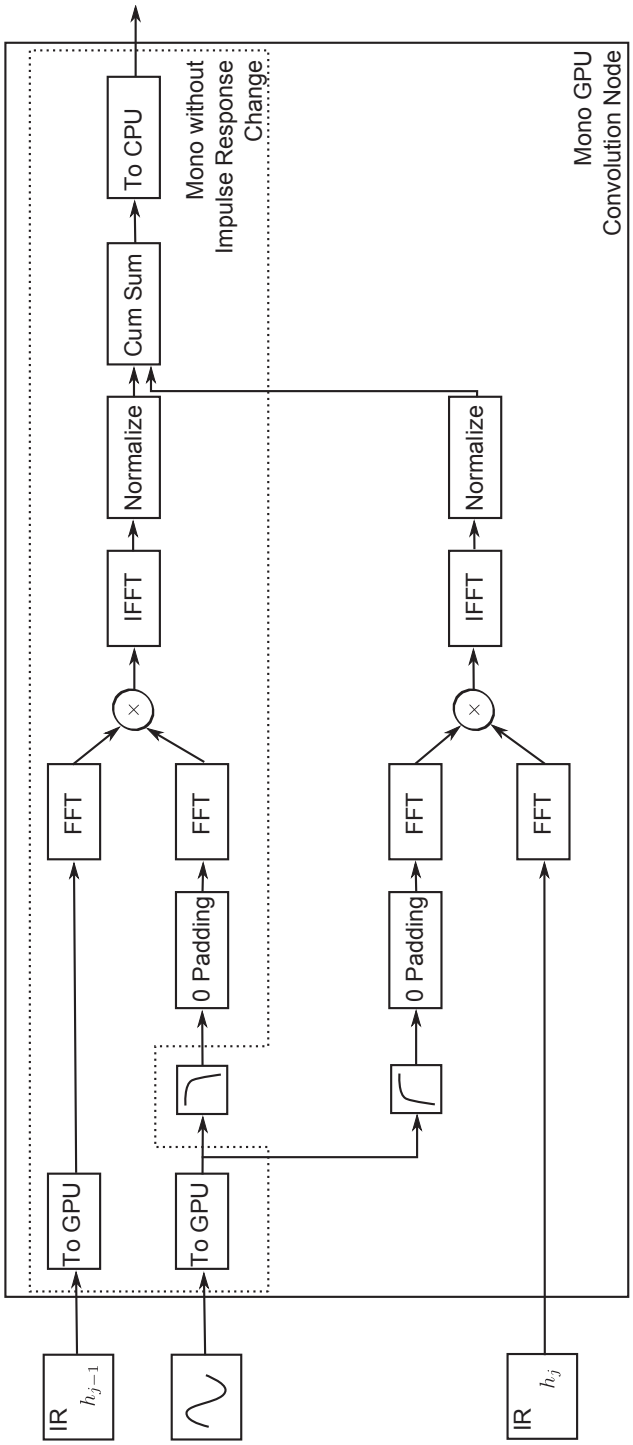


Figure 5.4: GPU convolution audio node.

Figure 5.4 shows the graphical representation of this transform. The steps recalled above represent the convolution of a single block of audio signal without changes of the RIR. This part is summarized schematically in Figure 5.4, where the dotted line represents the part of the process performed when the impulse response is static.

In the case of dynamic updates of the RIR, the convolution is performed twice. First, the input block is windowed (fade out) and convolved with the previous RIR. Second, the block is windowed (fade in) and convolved with the current RIR. The rest of the process remains the same.

Figure 5.4 shows that the GPU auralization process can be seen as an audio graph composed of audio nodes. However, modularity is often incompatible with an optimized code. Some informal tests were conducted on the implementation of the convolution algorithm by different audio nodes, but to improve the performances of our algorithms, the process described previously was implemented as a single audio node.

It is important to note that the procedure presented in this paragraph implements monophonic convolution. For stereophonic broadcast, the same process must be duplicated — using the optimization presented in Section 3.3.2.

5.3 Multi-thread general structure

We have presented in the previous section a method to perform parallel operations of DSP. We presented the structure of *audio graph*, and the choice that we made to provide a parallel calculation of audio nodes. However, DSP only represents a part of the total auralization framework. In this section we present the choices that were made to improve the parallelization of the algorithm.

The major problem that arises when implementing parallel algorithms is the sharing of data between concurrent processes. Imagine two audio nodes (A and B) generating audio at the same time and writing to the same memory place. The first node that finishes its process (node A) will write data in memory. Just after, as the second node finishes (node B), it will overwrite the values generated by A. Imagine now a third node (node C) that reads the values asynchronously at the same location in memory. Three behaviors can occur; (i) if C starts to read just after A finished writing, then it will read the values from A. (ii) If C starts to read the values just after B finished to write, then it will read the values from B. (iii) If C starts to read while A or B are writing, then C will read incoherent values that are a mix of A and B values. This last case is problematic in audio processing, because it produces audible artifacts. It is also a problem for all parallel computing structures, because incoherent values will lead to undesired behaviors.

One of the solution to prevent this concurrent access to the data is to implement *mutual exclusion*[9] mechanisms on the shared data. *Mutual exclusion* is a method that locks and unlocks processes. For instance, in the previous example of the two writers and the reader nodes; all of the three processes can use a mutex to lock the access to the memory block where the shared data are read/written. If A and

[8] *Object*, here refers to object oriented programing.
[9] *Mutual exclusion* are often called mutex.

C try to access simultaneously the same block of memory, the first will acquire the mutex. The second will wait until the first has finished (until the mutex is unlocked) to start its processing (and acquire the mutex).

The major drawback with this method is that a critical section[10] can only be executed by one thread at a time. Thus, the behavior of the application becomes sequential, as the threads are waiting for the critical section to be released. So, to be optimal, the mechanism of mutual exclusion should be used in very simple operations (such as vector swapping) in order to reduce to the minimum the waiting time of the processes. We present in the following section some memory management methods used to share efficiently memory between processes.

5.3.1 Multiple buffering

Buffers which are blocks of memory are intensively used in audio processing. The *audio buffer* represents the element that transits between each audio node. In a more general context, a buffer can be seen as a chunk of memory used to read, write or share data. In our implementation, we deal with audio buffers, buffers of rays, buffers of impulse responses or buffers of echograms.

Circular buffer

One type of buffer that is very often used in audio processing is the circular buffer. A circular buffer is a part of memory where the data are written circularly, *i.e.*, when the write pointer reaches the end of the structure, writing continues from the beginning, thus, updating the values previously written. This structure is for instance implemented in the delay lines presented in Sections 1.2.7 and 3.2.2. It is well suited to the delay line, as there is a distance (the delay) between the writing and the reading pointers. Thus, there is no risk that the reading and writing pointers overlap unless the delay is zero or the delay is greater than the size of the circular buffer. For these reasons, two constraints were imposed on the delay lines: delays must never be smaller than 3ms, *i.e.*, the propagation of sound over one meter in the air, nor greater than the maximal size of the RIR. Thus, the size of the delay line is defined at the beginning of the application as the size of the Room Impulse Response (RIR) generated.

Double buffer

We have presented in the introduction of this section the problem of two writers and one reader accessing the same chunk of memory. This example aims at presenting the problem of concurrent accesses. Generally, a good conception tries to avoid as much as possible that two processes write on the same memory zone. Instead, there are often more than one process that reads the results of one process, *e.g.* the audio node that reads sound files sends its content to both the binaural spatialization and GPU convolution modules.

One memory structure is well suited to the problem of one writer and many readers. It is called *double buffer* or *double buffering*. It has been extensively studied in the field of computer graphics. We

[10] A part of the code that usually has access to shared data.

first present its usage in computer graphics, in order to have concrete examples of its usage. Then we will extend it to the structure of triple buffer, and its implementation in our application.

In 2D or 3D graphics applications, double buffer is a structure where the data is prepared in a first buffer (that has the same dimensions as the image to display on screen). Once the data are prepared, the buffer is swapped with a second buffer. The first buffer thus contains the last prepared data, it will be sent to the screen for visualization. The next writing process will thus be executed on the second buffer while the first is displayed on screen.

The problem with this method is that a buffer must be locked (with a mutex) as long as it is read. In the case of many readers accessing the same memory chunk, or reading operations having a long access to the buffer, this solution may block the writing process unnecessarily.

As a solution, we propose a method based on triple buffers in the next section. This method is first presented in the framework of computer graphics to keep the analogy with this paragraph, and then extended to more general buffers.

Triple buffer

Triple buffer, as its name indicates, is analog to the method presented in the previous section with a third buffer. With this method, there is always a buffer free for writing while one is being read by another process. To present this structure, we reuse the example of the images produced in computer graphics that are sent to the screen. Imagine that the process that generates the images finishes his work while the screen is reading the other buffer. With double buffering, the writing thread has to wait for the end of the reading thread before starting to generate a new frame. With triple buffering, the buffer that is currently read is locked, so, it is impossible to permute it. But, the writing thread can start to process the new data in the third buffer that is free at this time. So, with this method, none of the threads get locked for more than a pointer swap operation (a very short operation).

In our developments, we have used this method for all the exchanges of data between threads. So, we implemented triple buffering using a template structure[11] in order to use triple buffers of rays, echograms and impulse responses. Figure 5.5 shows the full auralization process with triple buffering.

5.3.2 Task scheduling

All the modules of the auralization framework presented in Figure 5.5 are processing in parallel. As for audio nodes, the different modules are implemented as *active objects* [*Foote and Yoder*, 1998]. An additional module was developed called *the pilot*. The aim of the pilot module is to schedule the tasks of the other modules, and to interact with the elements outside the auralization framework. The interactions presented as a *control interface* in Figure 5.5 could be a Graphical User Interface (GUI), or another application that pilots the auralization framework.

While the source or the receiver moves in the virtual scene, *pilot module* collects the related information, and transmits it to all other modules. The next section explains how the modules are updated

[11] Template refers to object oriented development [*Stroustrup*, 1997].

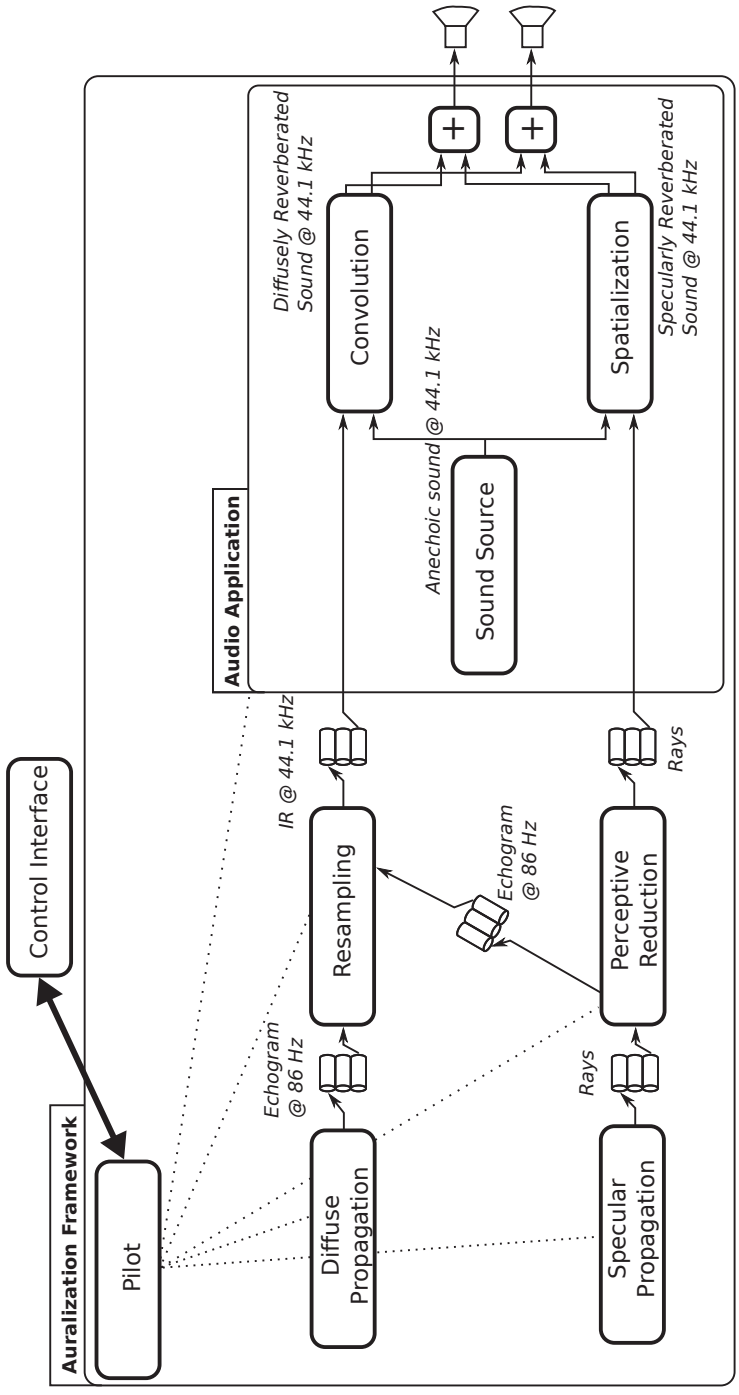


Figure 5.5: Full auralization framework with triple buffering and the pilot module.

during the displacement of any element in the virtual scene.

5.4 Progressive impulse response update

The computational structure we presented in the previous sections permits to update dynamically all the algorithms when either the source or the receiver moves. However, the calculation of the full propagation for both specular and diffuse fields can not be performed in real-time with the algorithms presented in Chapter 2. With this implementation, we had a calculation of the order of twelve seconds to perform the full update of the specular paths up to order fifty and the diffuse field up to order two hundred in the scene described in Appendix B.1.3. Thus, we implemented an incremental mechanism that updates the propagation elements incrementally during the auralization process. The principle of these updates is that the most significant elements are always updated first in order to have a processing time above 11.6 ms.

5.4.1 The progressive update algorithm

Incremental update of the propagation algorithms

We recall from Section 2.5 that we defined an original ordering for the propagation of rays and particles for the Deterministic Ray Tracing (DRT) and Monte Carlo Ray Tracing (MCRT) algorithms respectively. Usually, in ray or particle based algorithms, the elements are propagated until they are absorbed or until they fall below a certain energy threshold. Then, a new ray or particle is propagated. This is repeated until all elements are propagated.

In our implementation, all elements are propagated from the source until they intersect a wall. The position of the intersection point and the direction of reflection are stored, and all other rays are propagated for their first order of reflection. Once all elements are propagated, the algorithm continues for the second order of reflection and so on, until the rays or particles reach a given order of reflection, or they disappear from the virtual scene. This method gives a better control on the elements that are generated by the two propagation algorithms during runtime. At the cost of extra storage of the buffers for intermediate results.

The progressive update algorithm

As all of the modules that compose the auralization framework process in parallel, it is impossible to give a sequential representation of the algorithm. So, we will present the major principles that compose the progressive rendering algorithm. The details about execution times and orders of reflections will be presented in the next chapter.

Pre-computation When the auralization framework is started, the source and the receiver are assigned to a given position. An initial computation step is performed in order to build the full propagation

for both specular and diffuse fields at these positions. This operation could also be operated as a pre-calculation, as the position of the source and receiver are rarely different at start time for a given scene. The aim of this pre-calculation step is to obtain the full propagation for a given source-receiver position. This will be used in the subsequent steps.

Moving source and/or receiver When either the source or the receiver moves, the RIR needs to be updated. During a moving operation, only the first orders of reflections of the DRT and the MCRT algorithms are updated depending on the available computational resources. A good strategy to determine which orders have to be updated could be to adapt the rendering hierarchy principles presented in Section 2.4 with our propagation algorithm such as to yield an appropriate criterion for each algorithm of which orders can be updated. However, we did not re-implement this algorithm in the auralization framework. Instead, we defined the orders of reflections to update arbitrarily after informal listening tests and observations on the execution time of our algorithms. The observations on the orders of reflection and the execution times are provided in the next chapter.

Fixed source and receiver When the source and the receiver stops moving, the propagation of both specular and diffuse reflections algorithms is incremented to further propagation depth. As long as the source and the receiver stay in the same position, the propagation algorithms will continue the calculation for these positions. If the stop phase is as long as the pre-calculation phase, then, the information rendered by the auralization framework represent full propagation data associated with current positions. Before reaching this state, the rendering is a mix of the early reflections calculated at current position and the late reflections calculated at previous positions. However, we have seen in Section 2.4 that the late sound field in enclosed spaces is very coherent for late reflections. Thus, the difference between the sound field rendered, and the real sound field is small. This justifies the proposed approach.

Update of the other modules The modules of perceptive reduction and re-sampling are synchronized with the calculation of the propagation algorithms. When the calculation of one or more reflection orders by the propagation algorithm finishes, a new clustering and a new re-sampling are applied (asynchronously in different threads).

The module in charge of the audio rendering, the *audio application*, is not synchronized with the propagation algorithms. Instead, it is synchronized with the sound card clock. So, it gathers the last prepared information from the clustering and re-sampling modules. We see here the importance of the triple buffer structure, as it allows for instance the *audio application* to access to information provided previously by the *re-sampling* module while it is working on the preparation of new data. Figure 5.6 depicts the interactions between the modules while the source/receiver are moving, or when the sound card emits a clock signal.

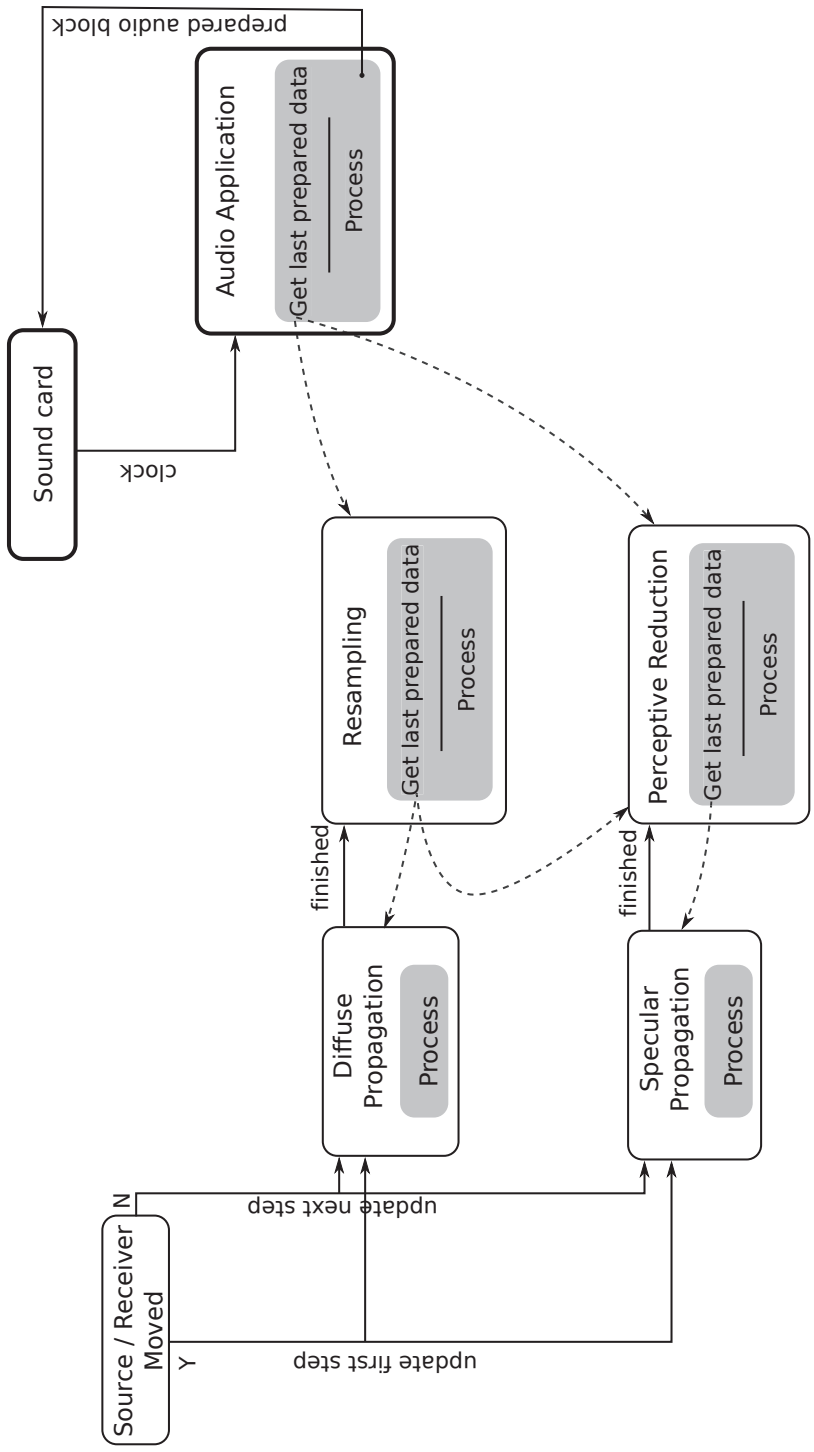


Figure 5.6: Incremental algorithm for the asynchronous update of the modules according to the movement of the source or listener.

5.5 Conclusion

At this stage, we have:

- two propagation algorithms that find all paths (specular and diffuse) between a source and a receiver in a virtual scene;
- a costly binaural auralization algorithm for pure specular paths;
- an efficient convolution algorithm on GPU for the rendering of the diffuse field, and the least significant specular paths;
- an algorithm that selects the most significant specular paths in real-time;
- a complete implementation in parallel on CPU and GPU;
- and an incremental method to concentrate the computational resources on the most significant parts of the auralization.

So, we have a dynamic sound rendering framework of complex environments. The next chapter is dedicated to the validation of the algorithms presented in the previous chapters.

6 Validation

The algorithms presented in Chapters 3, 4 and 5 were tested on the scenes of the third round robin in room acoustics [Bork, 2005b]. In Section 6.1, we present the different test procedures used for the validation of our algorithms. We have presented in Section 1.4 objective parameters used in room acoustics to evaluate the quality of an enclosed space. In Section 6.2.1 we compare those parameters with the other auralization software that participated in the round robin test [after Bork, 2005b]. In Section 6.2.2, we present the execution times of the different modules, in order to validate the use of our framework for real-time auralization. Finally, the strengths and weaknesses of our algorithms are discussed in Section 6.3.

Contents

Parameter	OL	RT
N_{Rays} — Specular	10.000	10.000
$N_{particles}$ — Diffuse	100.000	100.000
Depth Specular paths	50	21(1,2,3,15)
Depth Diffuse paths	200	140(5,10,15,20,25,30,35)
Cluster Δt	10^{-3} s	10^{-3} s
Cluster Δt_{inc}	10^{-6} s	10^{-6} s
Cluster ΔL	21 dB	21 dB
Nb of specular paths rendered	500	50

Table 6.1: Parameters for Off-Line (OL) and Real-Time (RT) test procedures.

6.1 The test procedure

The tests were performed on the three scenes described in the round robin in room acoustics [*Bork*, 2005a,b]. The geometric data of the scenes are presented in Appendix B. The first phase of the round robin consists in a very simple scene with seven walls, with absorption and diffusion coefficients equal to 0.1 for all frequencies. The three test scenes represent the studio of the *Physikalisch-Technische Bundesanstalt* (PTB) with different levels of details. The scenes of the round robin in room acoustics are defined with polygons. Our propagation algorithms uses a fast ray intersector on CPU developed by *Segovia* [2007]. This intersector takes as input a triangulated scene, and stores it in a kd-tree [*Wald and Havran*, 2006]. So, the first part of the processing consists in splitting the polygons into triangles to feed the intersector. The three triangulated phases of the round robin have respectively 16, 180 and 632 triangles.

Two series of test were performed. The first focuses on Off-Line (OL) rendering. The aim of this phase is not to reach real time rendering, but to focus on the quality of the auralization. These tests are performed to obtain a reference in terms of quality of our algorithms, and in particular the propagation algorithms. The second serie of tests focuses on Real-Time (RT) rendering. The parameters of the algorithm are set to reach an execution time below 11ms per sample block for the audio rendering, and to have a small propagation time (below 50ms) for the first order of reflection. Table 6.1 shows the parameters used for the two test procedures.

The parameters of the clustering module ($\Delta_t, \Delta_{tinc}, \Delta_L$) are the same for the real-time and the off-line tests, as well as the number of rays, N_{rays}, and particles, $N_{particles}$. In the off-line test serie, the maximum depth of the specular and diffuse algorithms were set to 50 and 200 respectively. For real-time rendering, the propagation is performed step by step. When the source or the receiver move in the scene, one order of specular reflection and five orders of diffuse reflections are updated. When the motion stops, after the end of the previous steps of propagation, two additional specular orders and ten diffuse orders are calculated. The order sequence used for the tests is indicated in brackets in

Table 6.1. Finally, the binaural spatialization module renders 500 paths for the off-line tests, and only 50 for real-time tests. The audio blocks are always composed of 512 samples, and sampled at 44.1 kHz.

The real-time tests were performed on the phase two and three of the round robin, with closed curtains. The off-line tests were performed on the three phases (also with close curtains). Every test was performed for the six arrangements of microphones[1] described in Appendix B.1.

6.2 Results

In this section, we present the results obtained with our method for the different test cases. In a first section, we present the validation of the algorithms with the objective parameters of room acoustics presented in Section 1.4.3. Then, we present the execution time of the different modules that compose our auralization system, and present how real-time rendering is possible. The results of the tests will be discussed in the next section.

6.2.1 Validation

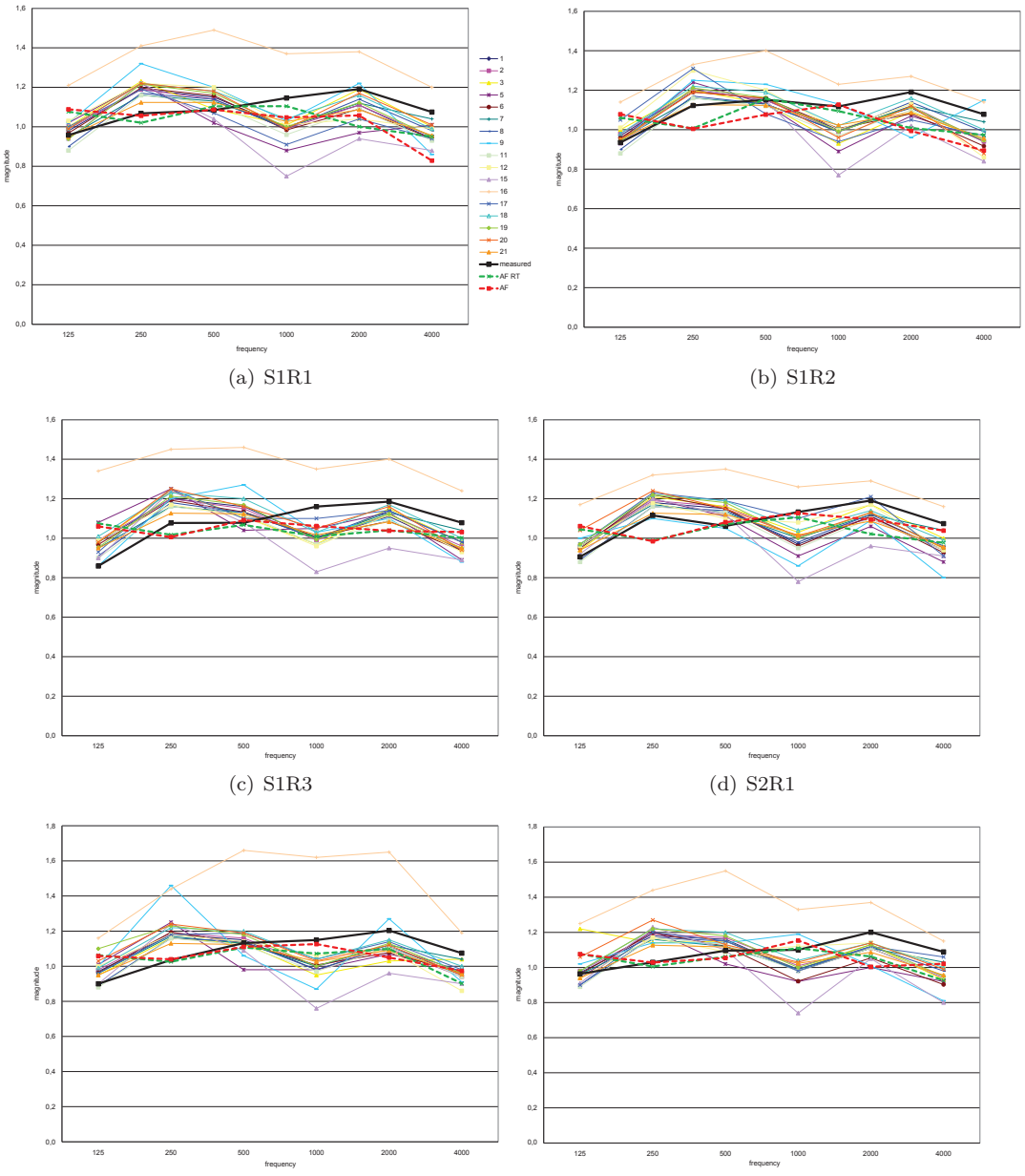
The results of the simulation for the second phase of the round robin with closed curtains are presented in this section. The results are compared with those of the other participants to the round robin. The participants numbered one to twenty one represent the other participants [*Bork*, 2005b], the bold black line represents the values measured in the real PTB studio [*Bork*, 2005a], AF represents our auralization framework with off-line parameters and $AF\ RT$ our auralization framework with real-time parameters. Figure 6.1 shows the reverberation time T_{30} obtained by all participants for all frequencies for all the positions of sources and receivers. Figure 6.2 depicts the four objective parameters T_{30}, EDT, C and D obtained by all participants for all frequencies at position S1R1. Figure 6.3 shows the average absolute value of the difference between calculation and measurement for the four objective parameters. To be consistent with the results presented by *Bork* [2005b], the results are normalized to the corresponding subjective limen listed in Table 6.2.

The extraction of the parameters was performed with the help of a software developed at CSTB, CritJan. This software was validated with the results of the *international round robin on room acoustical impulse response analysis software* [*Katz*, 2004].

6.2.2 Execution time

The second part of the results is dedicated to the execution time of the different modules of the auralization framework. The time of execution was calculated for both Off-Line (OL) and Real-Time (RT) rendering for the phases two and three of the round robin in room acoustics. It was also calculated

[1] two sources times three receivers denoted (S1R1,S1R2,S1R3,S2R1,S2R2,S2R3).



(a) S1R1

(b) S1R2

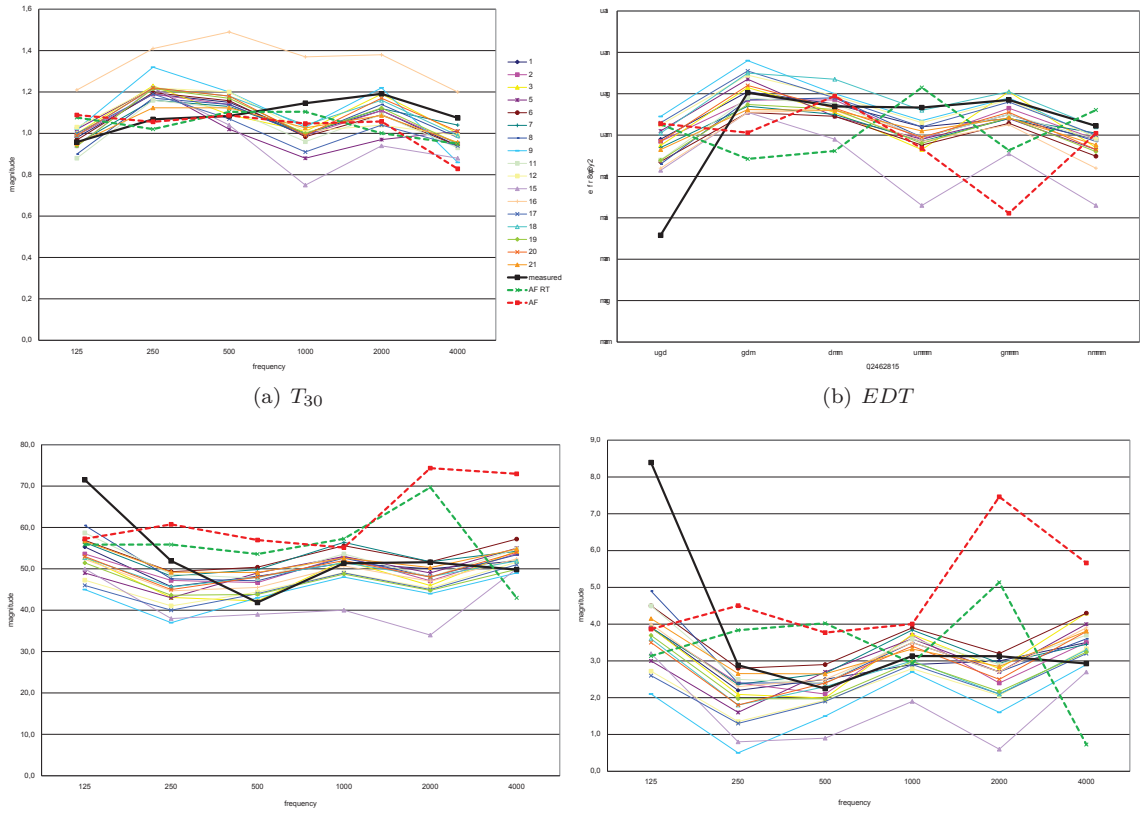
(c) S1R3

(d) S2R1

(e) S2R2

(f) S2R3

Figure 6.1: T_{30} for all arrangements of source receivers for the second phase of the round robin (with open curtains).



(a) T_{30} (b) EDT

(c) D (d) C

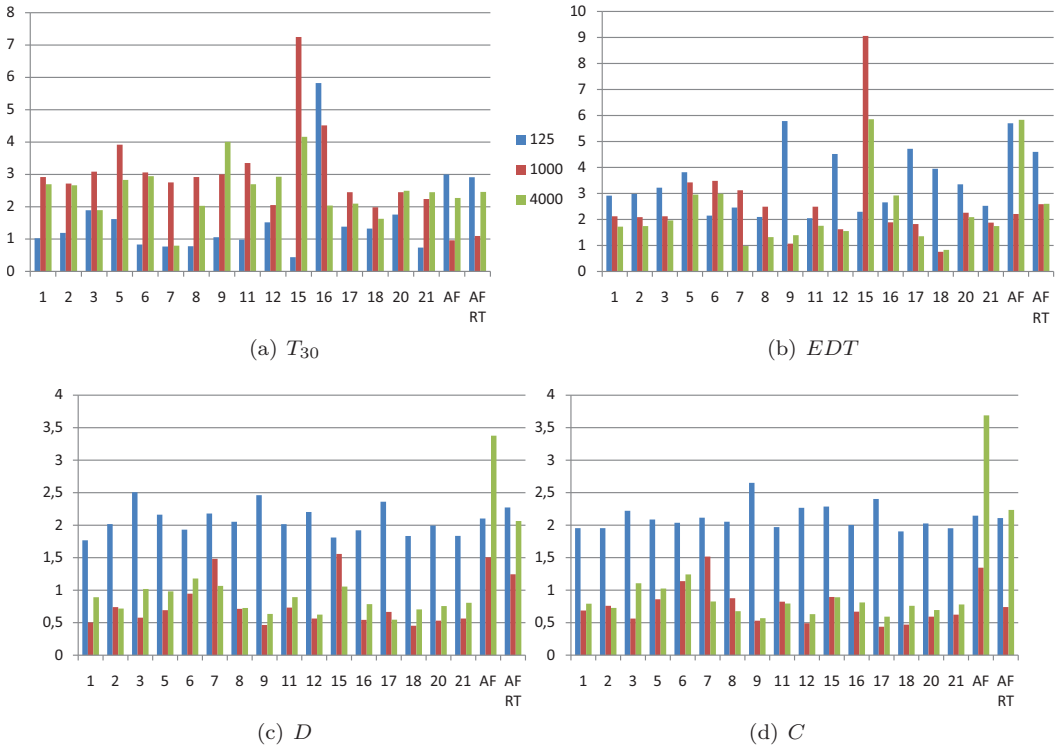
Figure 6.2: Objective parameters for position S1R1 for the second phase of the round robin (with open curtains).

parameter	subjective difference limen
T_{30}	50 ms
EDT	50 ms
D	1 dB
C	5%

Table 6.2: Subjective difference limen used as reference for Figure 6.3 [after *Bork*, 2005b].

	Phase 1	Phase 2 RT	Phase 2 OL	Phase 3 RT	Phase3 OL
Specular Prop.	191	172	310	278	505
Diffuse Prop.	11 913	11 193	16 331	11 972	16 911
Re-sample	14 427	13 815	13 969	13 344	14 021
Cluster	433	216	466	154	461
Audio-App	16 610	2 138	16 521	2 138	16 529

Table 6.3: Total execution time in milliseconds of each module for the five test cases.



(a) T_{30} (b) EDT

(c) D (d) C

Figure 6.3: Relative mean error of all participants for the second phase for three octave bands, averaged over six positions (with open curtains).

	Phase 1 OL	Phase 2 RT	Phase 2 OL	Phase 3 RT	Phase 3 OL
Specular Prop. (one order)	3,8	8,2	6,2	13,3	10,1
Diffuse Prop. (one order)	59,6	79,9	81,7	85,5	84,6
Audio-App (one block)	64,9	8,4	64,5	8,4	64,6

Table 6.4: Average execution time for one reflection order of the specular and the diffuse reflections algorithms, and one audio block processing by the audio application module.

for the phase one with OL rendering. The parameters used for the simulation are those presented in the previous section (*cf.* Table 6.1). The tests were conducted for the generation of a Room Impulse Response (RIR) of $2^{17} = 131072$ samples. The sampling rate is fixed at 44.1 kHz, thus, the impulse response has a duration of 2.97 seconds.

Let us remind that the tests were performed on an Intel Core 2 Quad at 2.66 GHz with 4 Gb of Ram and two NVidia 8800 GTX GPUs[2] .

Table 6.3 presents the execution times of all the modules presented in the previous chapters. These execution times are expressed in milliseconds. They were averaged over the execution time at the six sources/receivers positions. The test was executed once for each position, and the source and receivers were static. In order to simplify the comparison of execution times, Table 6.4 shows the propagation times for one order of reflection for both specular and diffuse fields. It also shows the audio processing time for one audio block of 512 samples. It is important to note that in these results, there is nearly one order of magnitude between the propagation time of specular reflection and diffuse reflection. This is due to the fact that there are ten times more rays/particles thrown for diffuse field than for specular reflections. These execution times were calculated directly from the values of Table 6.3 and divided by either the number of reflections or the size of the impulse response (in audio blocks).

6.3 Discussion

Starting from the results presented in the previous section, we discuss the strengths and weaknesses of our auralization framework, compared to other auralization systems presented in the bibliography. The discussion is split in two parts with first an analysis of the room impulse responses produced by our system, and then, the analysis of the execution times in order to reach real-time rendering.

Validation We have presented in Figure 6.1 the reverberation time, T_{30} for the six arrangements of the sources and the receivers in the scene, and in Figure 6.3(a) the average value for those six positions. Reverberation time is certainly the most important parameter in the field of room acoustics. It is

[2]Curent implementation only uses one GPU. The second can be used for instance to implement propagation algorithms or the Binaural Spatialization Algorithm (BSA).

the first one calculated by the acoustician, and the acoustical quality of an enclosed space is at least expressed depending on its reverberation time. We see, from the results presented in Figure 6.3(a), that our estimation of the reverberation time is within the 0.15 ms of the measured reverberation time. This result is equivalent to the results of the other participants of the round robin for frequencies above or equal 1 kHz. The results found for the lower frequency band are worse than those of the other participants except number sixteen. This problem in low frequencies will be discussed later in this section. The other parameters studied in Figures 6.2 and 6.3 are more sensitive to small variations in the impulse response. For the EDT, we obtained results equivalent to participants nine to twenty one, but worse than participants one to eight. From the results expressed in *Bork* [2005b], we suppose that these results are extracted from commercial softwares. For the calculation of parameters D and C, we obtained worst results with our algorithm. We recall from Section 1.4.3 that those two parameters are very sensitive to the fluctuations of impulse response around 50 and 80 ms respectively. In our algorithms, there is a great influence of the clustering algorithm at the beginning of the impulse response.

Some informal tests have been conducted in order to find the drawbacks of our method and to explain the possible causes of these errors on C and D parameters. First, the tests have been conducted without the perceptive reduction module. The 1000 first specular contributions have been kept and the others have been added to diffuse part of the echogram. Then, the integration period of the diffuse field has been changed between various values (see Section 1.3.3) in order to have a more precise diffuse response. Many other tests have also been performed on the basic propagation algorithms. They have been tested on simple test scenes (only one source and a receiver, a source a receiver and a wall, a shoe box room ...). None of these tests were sufficient to obtain a correct match of the criteria with those of the round robin.

Regarding the low frequency mismatch mentioned above, *Bork* [2005b] addresses this problem in the conclusion of his paper. He states that the limits of the methods used for propagation have been reached. According to him, the only way to improve the calculation in the low frequency region is to consider finely the diffraction of the sound wave, the complex impedance of the materials and the interferences of the waves. Thus using *wave based methods* such as *BEM* seems to be a good way to correct these weaknesses. However, as of today, those methods are not yet suited to real-time auralization.

In order to validate the spatial effects in our algorithm, the other criterion of the round robin in room acoustics [*Bork*, 2005a] should be studied. The three criterions Interaural Cross-Correlation (IACC), Lateral Fraction (LF) and Lateral Fraction Cosine (LFC) give hints on the spatial validity of room simulation algorithms. In this thesis, the efforts have first been focused on the validation of the criterion presented earlier. The study of the spatial criterion is left for a further study.

Another aspect that emerged during the analysis of our algorithms is the variance of the results. In order to continue the validation of our algorithms, an additional study should be performed, focusing on the parametric evaluation of all modules. The parameters defined for the simulations in this chapter were defined more or less arbitrarily. In room acoustics software, there are often pre-defined parameters such as the depth of the propagation algorithm and the number of rays/particles to throw. But, the choice of these parameters is more often guided by the experience of the acoustician who runs the

software. Note that this is consistent with the results presented by [*Bork*, 2005b] where different users using the same software to process the scenes of the round robin obtain different results. The extended study of our algorithms should include the parametric study of every parameter presented in Table 6.1, their influence on the calculation mean value, variance, and execution time. We have already proposed in Section 2.4 a method to determine the hierarchy of the diffuse and specular process. This method was tested in the case of simple scenes with the mixed image sources/radiosity algorithms. A good way to start this parametric study would be to implement this algorithm for Deterministic Ray Tracing (DRT) and Monte Carlo Ray Tracing (MCRT) algorithms.

Execution time A short observation on the execution time of the modules in Table 6.3 yields to the conclusion that it is impossible to execute the full auralization process in real-time. Real-time processing implies a processing time inferior to the signal generated (here 2.97 seconds). With the real-time parameters, the execution time of the diffuse reflection algorithm and the re-sampling take respectively 11.2 and 13.8 seconds for the second phase of the round robin. However, it is possible to execute the audio application module in real-time. According to Table 6.4, the execution time for one audio block with RT parameters is 8.4 ms which is inferior to the block duration (11.6 ms). Table 6.4 also shows that the execution time of the audio application is very stable for both RT and OL configurations. It is important to have a low variance of the execution time of the real-time algorithms, as a high variance would lead to un-calculated blocks and thus audible artifacts.

In this thesis, we present a solution to perform all the parts of the auralization in real-time, not only the audio rendering. The solution presented in Chapter 5 is the progressive impulse response update, it aims at splitting the computation time of the propagation algorithms. Instead of performing the full propagation for both specular and diffuse fields, the propagation is performed step by step. A step consists in multiple orders of reflection. Table 6.1 gives the sequences of orders used for real-time rendering in our auralization framework. With this method, the propagation of the most significant paths is performed while the source or the receiver moves in the virtual scene. When they are static, the propagation is extended to further propagation steps, until all specular and diffuse paths are updated. According to Table 6.3, when the source and the receiver are static for 278 ms, all specular paths are updated. When they are static during twelve seconds, all the impulse response is updated to the new position.

We have seen earlier that the propagation intersector uses a kd-tree hierarchy. Table 6.4 shows that the computation time for the intersectors as a sub-linear complexity related to the number of triangles in the scene. This is a good point for the application of the proposed algorithms to complex scenes.

In Table 6.4, we observe that the computation time per reflection order is nearly always longer for real-time computation than for off-line computation. This is mainly due to the fact that in the off-line tests, the propagation is performed in one step (all orders of reflection at once). For real time processing, the mechanism that spreads the computation in different propagation steps involves the storage of the propagated rays or particles at each step. Thus, it consumes more CPU time.

The main drawback of our current implementation of the algorithm is the re-sampling module. It

has a long execution time, and needs to be executed every time the diffuse part of the signal is modified. As this module represents the link between the propagation and the audio application, it needs to be executed before the auralization of diffuse field operates on the latest diffuse information. According to Table 6.3, the re-sampling takes around fourteen seconds to create the diffuse stereo echograms used for the convolution. Thus, the auralization of the sound at time t is performed with the diffuse reflection calculated at time $t - 14$ seconds. Fortunately, the diffuse field changes slowly in enclosed spaces, and this delay can be tolerated. However, in order to have a more accurate rendering system, this module should be optimized. The first optimization we could imagine would be to perform a progressive update of the diffuse impulse response. Currently, when a part of the diffuse field is modified, the re-sampling module is executed to recreate the whole diffuse impulse response, even if only a small part of it has to be recalculated. Further, when applying the progressive impulse response method (when the source or the listener move), the modified part of the impulse response is always located at the beginning. Thus the re-sampling could be performed only on this part. As of today, the major code optimizations were performed in the module *audio application*, as real-time execution of this module is essential for artifact free auralization. Future work will involve optimization of the re-sampling module.

* * *

In this chapter we presented the two test procedures that were used to validate the algorithms presented in the previous chapters. The first validation focuses on the quality of the Room Impulse Response (RIR) rendered by our algorithm. Our auralization framework is compared to the solutions of the other participants of the third round robin in room acoustics [*Bork*, 2005b]. The second test procedure focuses on the execution time of each module, and their arrangement for real-time auralization. Finally, the strengths and weaknesses of our algorithms were discussed, and some possible improvement directions presented.

Synthesis of the research and perspectives

The contributions presented in this document concern the creation of an auralization framework for *dynamic sound rendering of complex environments*. The results that were obtained thanks to improvements in all the elements of the auralization process.

A great variety of propagation algorithms were developed since the middle of the 70's. We presented in Chapter 2 the most significant ones, with their main characteristics and finally chose two of them for the auralization of specular and diffuse sound field. In the present work, these algorithms were modified to satisfy the constraints of real-time rendering. We saw in Chapter 6 that the validation of these algorithms with the objective parameters of room acoustics is not as precise as commercial softwares based on off-line calculations. However, the comparison still shows promising results. In order to use our algorithms in room acoustics projects, a deeper validation of the algorithms should be conducted. Some of the directions to investigate were discussed at the end of Chapter 6.

Real-time is an important aspect in this study. Four contributions are related to this aspect:

- a binaural spatialization algorithm,
- a convolution algorithm on GPU,
- a thread structure to arrange all interconnected modules on both CPU and GPU,
- a progressive impulse response update method.

The auralization framework is designed as a set of interconnected modules. In order to produce dynamic sound rendering in real-time, the DSP process and the global computational structure have an efficient thread distribution were investigated and implemented. According to the results presented in Chapter 6, real-time requirements are fully operational on the test computer. With newer generations of computers, these limits can be pushed further. This newer generation would allow for instance the rendering of multiple sound sources, with only slight modifications of the framework structure.

Finally, the novelty of our approach, compared to most of the real-time auralization methods is to include a perceptive simplification between the propagation and the auralization stages. The accurate rendering of specular paths in real-time remains a research challenge. However, we have provided a fast algorithm to extract the most significant paths from a perceptive point of view, and thus to concentrate the calculation on those paths. The principles of this algorithm were presented, as well as a first estimation of the relevant parameters. However, in order to have an optimal rendering, further investigations should be conducted to finely tune those parameters according to the virtual scene characteristics. Indeed, subjective listening tests were conducted for only one room, and with an older version of the auralization framework. The first step toward a better estimation of the parameters should be to perform new subjective tests with the new auralization framework on scenes with various material properties. It is obvious that the influence of specular paths will be lesser in more diffusive rooms.

Some restrictions were fixed at the beginning of the study to concentrate on real-time rendering. One of them is the limitation of the propagation to specular and diffuse paths. This important constraint

restricts the validity of our algorithms to scenes where the diffraction has very little influence. Thus, it is impossible to have an accurate rendering in *e.g.* coupled volumes, or scenes like open spaces. In coupled volumes, the diffraction by doors is important. In specular models, the interaction between a source and a receiver is binary (and analogous to light propagation): if the source and the receiver see each-others, the path exists, otherwise, it is discarded. In acoustics, an occluded path still exists, but it gets attenuated. This attenuation is explained in the *geometrical theory of diffraction* [*Keller*, 1961]. This observation is true for all cases where the source and receiver are partially occulted. This is the reason for which our algorithm is not suited in its actual version to outdoor propagation, where most of the significant paths encounter diffracting edges. However, the implementation of diffraction in our framework only impacts the propagation phase. The other modules can be reused.

To conclude, we focused this study on high quality rendering dedicated to the perceptive and interactive evaluation in the field of room acoustics. It could be interesting to work on algorithms with a lower degree of fidelity to meet the constraints of audio rendering in other fields such as audio-visual production or video games.

A

Mathematical tools

Here are some of the mathematical tools that are useful for the comprehension of the document.

A.1 Solid angle [after *Holtzchuch*, 2009]

The solid angle is the extension to 3D coordinates of the standard angle in 2D coordinates. It is measured in steradian. For an element of area A on a sphere of radius r, the solid angle is:

$$\Omega = \frac{A}{r^2} \tag{A.1}$$

If we want to compute the solid angle of an object, as seen from a point, we integrate the differential solid angle, Ω, over all surface elements of the object, S (see Figure A.1):

$$\Omega = \int_S \frac{\mathrm{d}A^\perp}{r^2} \tag{A.2}$$

$A^\perp$, is the projection of A, and S is the surface of the object.

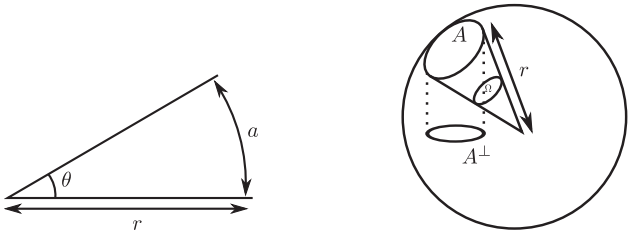


Figure A.1: The solid angle, Ω, on a sphere of radius r, is the ratio of the surface, A to r^2, just like the angle, θ, on a circle of radius, r, is the length of the arc a divided by r.

A.2 Spherical coordinates [after *Pharr and Humphreys*, 2004]

Spherical coordinates and Cartesian coordinates are the two 3D representation systems that are widely used in this document. The conversion from Cartesian (x, y, z) to polar coordinates (d, θ, ϕ) is[1]:

$$\begin{cases} d &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \cos^{-1}\left(\frac{z}{d}\right) \\ \phi &= \tan^{-1}\left(\frac{y}{x}\right) \end{cases} \tag{A.3}$$

the inverse conversion is:

$$\begin{cases} x &= d\sin\theta\cos\phi \\ y &= d\sin\theta\cos\phi \\ z &= d\cos\theta \end{cases} \tag{A.4}$$

A.3 Dirac distribution [after *Holtzchuch*, 2009]

The Dirac distribution, δ, also called Dirac delta function, has the following properties:

$$\begin{aligned} \int_I f(x)\delta(x)\mathrm{d}x &= 0 \text{ if } 0 \notin I \\ \int_I f(x)\delta(x)\mathrm{d}x &= f(0) \text{ if } 0 \in I \end{aligned} \tag{A.5}$$

The Dirac delta function is a *distribution*, it can be seen as the derivative of the Heaviside function. The Dirac distribution is also often used centered on a given point x_0:

$$\begin{aligned} \int_I f(x)\delta_{x_0}(x)\mathrm{d}x &= 0 \text{ if } 0 \notin I \\ \int_I f(x)\delta_{x_0}(x)\mathrm{d}x &= f(x_0) \text{ if } 0 \in I \end{aligned} \tag{A.6}$$

In this document, the Dirac distribution is used both in the fields of Digital Signal Processing (DSP) (see *e.g.* Section 1.2.7) and propagation (see *e.g.* Section 2.1.3).

[1] Note that spherical coordinates are often written (r, θ, ϕ). In this document, r, represents the radius of a sphere, so, we used, d, as the distance.

A.4 Expected value, variance, standard deviation [after *Pharr and Humphreys*, 2004]

The expected value, $E_p[f(x)]$, of a function, f, is defined as the average value of the function over some distribution of value, $p(x)$, over its domain. Expected value over a domain, D, is defined as:

$$E_p[f(x)] = \int_D f(x)p(x)\mathrm{d}x \tag{A.7}$$

When the distribution used is a uniform distribution, the subscript, p, from, E_p, is dropped, thus, the *expected value* is generally denoted E.

The variance, V, of a function is the expected deviation of the function from its expected value. Variance is a fundamental concept for quantifying the error in the value estimated by Monte Carlo algorithms (see Section A.7). The variance is defined as:

$$V[f(x)] = E\left[(f(x) - E[f(x)])^2\right] \tag{A.8}$$

Another equivalent formulation that is commonly used to compute the variance is:

$$V[f(x)] = E[f(x)^2] - E[f(x)]^2 \tag{A.9}$$

Thus, the variance is the expected value of the square minus the square of the expected value. The standard deviation, ρ, is defined as the square root of the variance:

$$\rho(f(x)) = \sqrt{V[f(x)]} \tag{A.10}$$

A.5 Interpolation [after *Niemitalo*, 2001]

In this section we present different interpolators after *Niemitalo* [2001]. The original article is focused on re-sampling of audio signals, but the remarks about the interpolators formulated in this article can be applied to every type of 1D signals.

A.5.1 Drop-sample interpolator

The *drop sample*, also called *0th-order*, *0th-order B-spline* or *nearest neighbor* interpolator is the easiest way to to perform the interpolation of a 1D signal. The principle is to take the nearest value of input signal and keep it constant in the output signal.

The impulse response of the *drop sample* interpolator is:

$$f(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Figure A.2: Drop-sample interpolation impulse response [after *Niemitalo*, 2001].

A.5.2 Linear interpolator

Linear interpolation, also called *1st-order interpolation*, is defined as the autocorrelation of the drop-sample interpolator. This is one of the most used interpolators as it suppresses the discontinuities of the drop-sample interpolator. The drawback of this interpolator is that its first derivative is not continuous. Thus leading to discontinuities in the first derivative of the signal.

The impulse response of the linear interpolator is:

$$f(x) = \begin{cases} 1-x & 0 \leq x < 1 \\ 0 & 1 \leq x \\ f(-x) & \text{otherwise.} \end{cases}$$

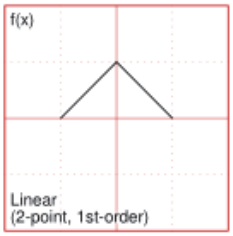


Figure A.3: Linear interpolation impulse response [after *Niemitalo*, 2001].

A.5.3 Hermite interpolator

Hermite interpolator, also known as *Catmull-Rom spline interpolator* is one of the most used interpolators for audio signals. His first derivative is continuous. The impulse response of Hermite interpolator is:

$$f(x) = \begin{cases} 1-\frac{5}{2}x^2+\frac{3}{2}x^3 & 0 \leq x < 1 \\ 2-4x+\frac{5}{2}x^2-\frac{1}{2}x^3 & 1 \leq x < 2 \\ 0 & 2 \leq x \\ f(-x) & \text{otherwise.} \end{cases}$$

Figure A.4: Hermite impulse response [after *Niemitalo*, 2001].

* * *

The reader may refer to *Niemitalo* [2001] for a complete review on more than thirteen interpolators, and the design of optimal interpolators.

Figure A.5 shows a signal interpolated with the three methods presented in this section.

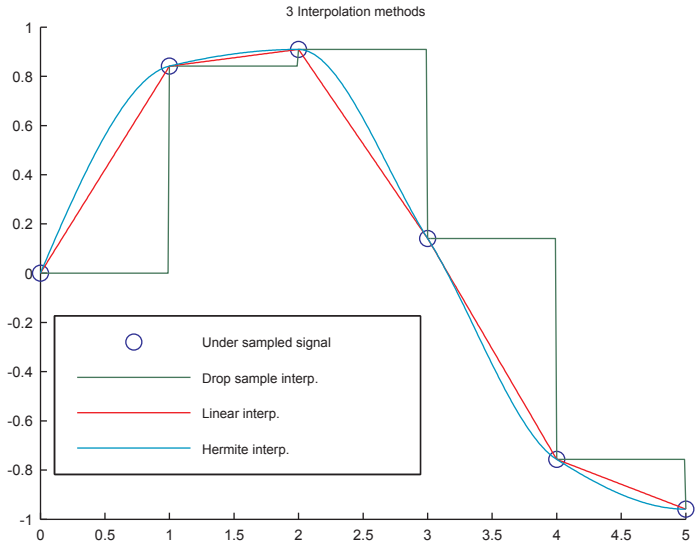


Figure A.5: Comparison of three interpolation methods (Drop-sample, linear and Hermite).

A.6 Numerical integration

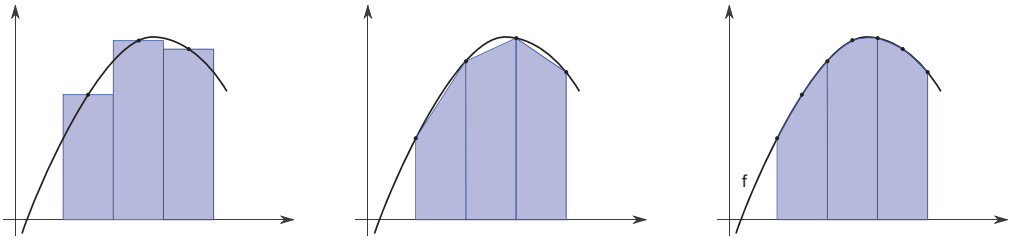
Numerical integration represents a broad family of algorithms for calculating the numerical value of an integral. Numerical integration is often called *numerical quadrature* for one dimensional signals. The problem set down is to find an algorithm to compute an approximated solution to a defined integral $\int_a^b f(x)\mathrm{d}x$. Usually, numerical integration methods are used when there is no analytical solution to the integral.

Usually, numerical integration is performed in three steps:

- Decomposition of the domain into contiguous sub-domains
- Approximation of the function for each sub-domain
- Summation of the numerical results

A.6.1 Rectangle method

The rectangle method is the easiest way to integrate numerically a function. Each sub-domain is integrated by a constant function (a zero order polynomial). The estimation of the integral can be



(a) Rectangle (b) Trapezoid (c) Simpson

Figure A.6: Three integration methods.

performed either on a, b or any point between a and b. Usually, we use the point located at $(a+b)/2$. Thus this method is called *middle point method*.

The value of the integral is:

$$\int_a^b f(x)\mathrm{d}x \approx (b-a) f\left(\frac{a+b}{2}\right) \tag{A.11}$$

A.6.2 Trapezoidal method

Trapezoidal method is a first order method that evaluates the integral on a sub domain with 2 points:

$$\int_a^b f(x)\mathrm{d}x \approx (b-a)\frac{f(a)+f(b)}{2} \tag{A.12}$$

A.6.3 Simpson method

Simpson method is obtained by interpolating f with a second order polynomial. Note that this general method works with all interpolators (including those presented in the previous section). Simpson formulation is:

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right) \tag{A.13}$$

Figure A.6 shows three representations of the methods presented in this section.[2]

A.7 Monte Carlo integration [after *Pharr and Humphreys*, 2004]

Monte Carlo (MC) is another integration method. Suppose that we want to evaluate a one-dimensional integral $\int_a^b f(x)\mathrm{d}x$. Given a supply of uniform random variables $X_i \in [a,b]$, the MC estimator is defined as:

$$F_N = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i) \tag{A.14}$$

[2] After http://fr.wikipedia.org/wiki/Calcul_num%C3%A9rique_d%27une_int%C3%A9grale.

The expected value of the estimator is equal to the integral we are computing:

$$E[F_N] = \int_a^b f(x)\mathrm{d}x \tag{A.15}$$

The main drawback of MC is its slow convergence rate for low dimensional functions: $\sqrt{N}$. The great benefit of MC estimator is its ease of implementation and the fact that the convergence rate is independent of the dimension of the integral.

However, some methods such as importance sampling, stratified sampling, or Russian roulette can be used to speedup the convergence of the algorithm. More information on MC can be found in *e.g.* *Millet* or *Pharr and Humphreys* [2004].

A.7.1 Russian Roulette [after *Pharr and Humphreys*, 2004]

Russian Roulette aims at improving the efficiency of MC algorithms at the cost of an increase of the variance in the result. Russian roulette addresses the problem of samples that are expensive to evaluate but makes a small contribution to the final result.

The Russian roulette is based on a termination probability q. This value can be arbitrarily chosen. For instance in the case of a ray reflecting on a surface, it can be chosen as the probability that the ray gets absorbed by the surface. With probability, q, the integrand is not evaluated for the sample. The sample is replaced by some value, c [3]. With probability, $1-q$, the integrand is still evaluated, but weighted by $(1-q)^{-1}$, that accounts for the samples that were skipped. The general formulation of the Russian roulette estimator is thus:

$$F' = \begin{cases} \frac{F-qc}{1-q} & \xi > q \\ c & \text{otherwise} \end{cases} \tag{A.16}$$

with, ξ, a uniform random variable in [0..1]. Note that the expected value of the resulting operator is the same as the expected value of the original estimator:

$$E[F'] = (1-q)\frac{E[F]-qc}{1-q} + qc = E[F] \tag{A.17}$$

The choice of a good Russian roulette is essential for a fast convergence of the simulation. For instance, a Russian roulette that discards too many *important* samples will increase drastically the variance of the simulation. Mathematically, the result will be correct, but the convergence will be very long.

[3] $c = 0$ is often used.

B

Test scenes

B.1 Round Robin 3 on Room acoustics

Most of the tests conducted on the auralization framework were performed on the scenes described in the third round robin on room acoustics [*Bork*, 2005a,b]. The scene used for the round robin was a studio of the *Physikalisch-Technische Bundesanstalt* (PTB). The first part of the report [*Bork*, 2005a] presents the measurement results for the room acoustical parameters according to *ISO3382-1* [2009]. The second part [*Bork*, 2005b] compares the results of numerical simulations submitted by a number of users and developers. In this document, we present the three phases studied in the article. The first phase is a simple model composed of seven walls representing approximately the geometry of the PTB studio. It was used to test the reliability of the software tested. Phases two and three present the scene of the studio with different levels of details. In both scenes, the tests were performed both with open and close curtains. The description of the scenes in different formats can be found at `http://www.ptb.de/en/org/1/16/163/roundrobin/roundrob3_1.htm`. Figure B.1 shows a photography of the PTB studio with wooden diffusers for a wall and the ceiling.

B.1.1 Phase 1

For this phase, a simple model was created with equal absorption and coefficients of all frequency bands ($\alpha = 0.1$ and $s = 0.1$). The shape corresponds to the shell structure of the studio. Figure B.2 presents the geometry of the scene, as well as the position of the sources and receivers. Table B.1 presents the position of the sources and receivers — These positions are identical for the three phases of the round robin.

Figure B.1: Musical studio of the PTB with wooden diffusers.

B.1.2 Phase 2

This is a more detailed model where the fine structure of the diffusing elements of the ceiling and the wooden wall is neglected and represented as two places with frequency-dependent absorption and scattering coefficient. The geometry is defined both with open and close curtains. Figure B.3 represents a view from our auralization software (auralization framework + GUI) of the studio for the second phase with open curtains. Tables B.2 and B.3 present respectively the absorption and scattering coefficients of the materials of the scene.

Id	x	y	z
S1	1.5	3.5	1.5
S2	-1.5	5.5	1.5
R1	-2.0	3.0	1.2
R2	2.0	6.0	1.2
R3	0.0	7.5	1.2

Table B.1: Position of the sources and receivers in Cartesian coordinates.

Material	$125Hz$	$250Hz$	$500Hz$	$1kHz$	$2kHz$	$4kHz$
parquet	0.04	0.04	0.07	0.06	0.06	0.07
wood	0.20	0.05	0.03	0.01	0.01	0.01
curtainc	0.15	0.30	0.35	0.40	0.50	0.55
curtaino	0.30	0.45	0.60	0.70	0.70	0.70
wilhelmi	0.42	0.28	0.49	0.78	0.58	0.62
studiowall	0.02	0.02	0.02	0.02	0.02	0.02
windowglass	0.10	0.04	0.03	0.02	0.02	0.02
woodabsorbers	0.40	0.33	0.21	0.16	0.15	0.16
ceiling	0.30	0.20	0.06	0.02	0.02	0.02

Table B.2: Absorption coefficients, α, for the materials of the second and third phase of the round robin.

Material	$125Hz$	$250Hz$	$500Hz$	$1kHz$	$2kHz$	$4kHz$
parquet	0.20	0.20	0.20	0.20	0.20	0.20
wood	0.20	0.20	0.20	0.20	0.20	0.20
curtainc	0.21	0.26	0.32	0.39	0.47	0.57
curtaino	0.22	0.31	0.39	0.48	0.59	0.73
wilhelmi	0.20	0.20	0.25	0.30	0.35	0.40
studiowall	0.20	0.20	0.20	0.20	0.20	0.20
windowglass	0.10	0.10	0.10	0.10	0.10	0.10
woodabsorbers	0.50	0.90	0.95	0.95	0.95	0.95
ceiling	0.13	0.56	0.95	0.95	0.95	0.95

Table B.3: Scattering coefficients, s, for the materials of the second and third phase (except woodabsorbers and ceiling) of the round robin.

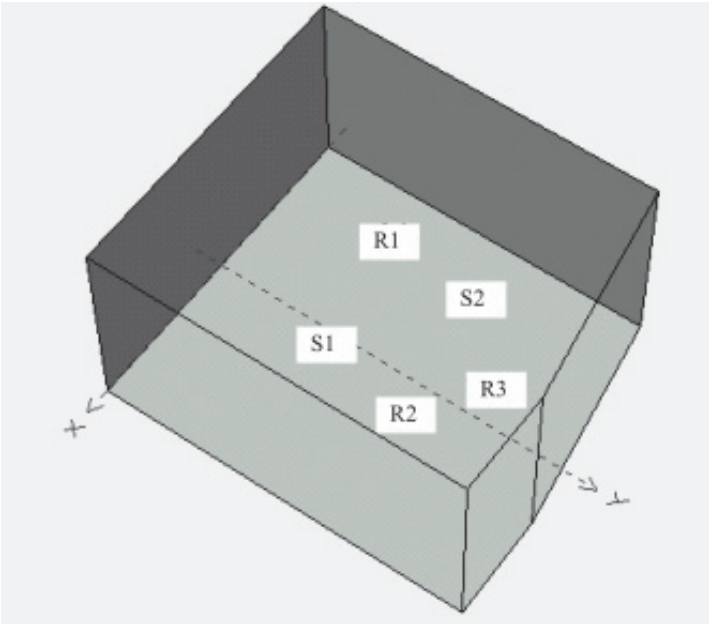


Figure B.2: The scene of the first phase of the third round robin on room acoustics with the position of the sources and receivers [after *Bork*, 2005b].

Material	$125Hz$	$250Hz$	$500Hz$	$1kHz$	$2kHz$	$4kHz$
woodabsorbers	0.20	0.20	0.20	0.20	0.20	0.20
ceiling	0.20	0.20	0.20	0.20	0.20	0.20

Table B.4: Scattering coefficients, s, for the materials of the third phase of the round robin.

B.1.3 Phase 3

This is the same model as the previous section with geometrical details added for the diffusing areas. Thus, the scattering coefficients are changed for the two materials as presented in Table B.4. Figure B.4 shows the detailed geometry of the diffusing wall and the ceiling. Figure B.5 represents a view from our auralization software of the for the 3rd phase with close curtains.

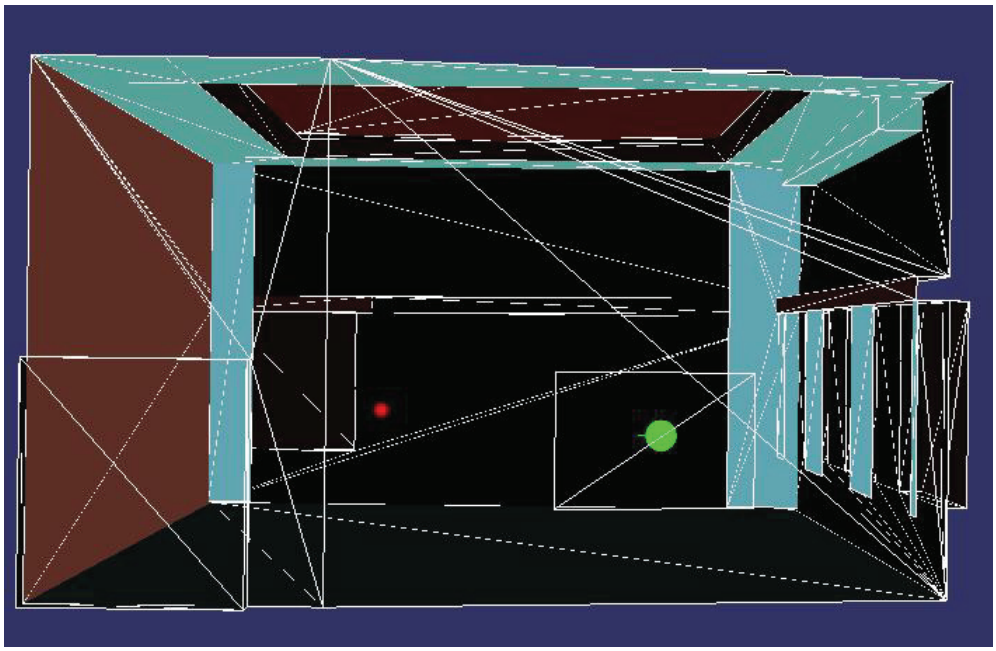
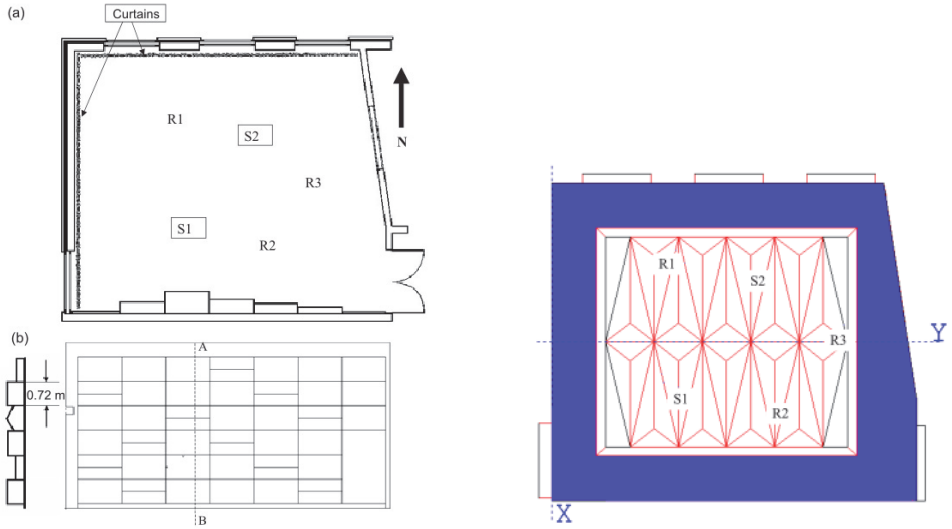


Figure B.3: The scene of the second phase of the third round robin on room acoustics with open curtains.



(a) Wall (b) Ceiling

Figure B.4: Details of the structure of the diffusing wall and ceiling of the PTB studio [after *Bork*, 2005a,b].

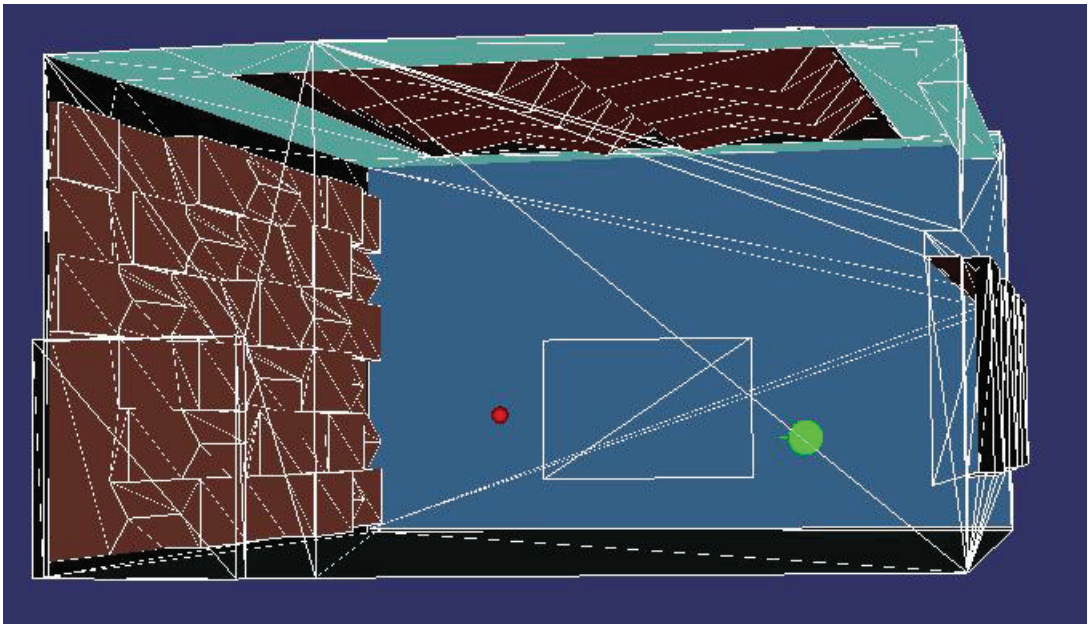


Figure B.5: The scene of the third phase of the third round robin on room acoustics with close curtains.

Bibliography

Allen, J. (1977), Short term spectral analysis, synthesis, and modification by discrete fourier transform, *IEEE Transactions on Acoustics, Speech and Signal Processing*, *25*(3), 235 – 238. 91

Allen, J., and L. Rabiner (1977), A unified approach to short-time fourier analysis and synthesis, in *IEEE Proceedings*, vol. 65 (11), pp. 1558 – 1564. 91

Allen, J. B., and D. A. Berkley (1976), Image method for efficient simulating small-room acoustics, *JASA*, *65*(4), 943–950. 45

Allred, J. C., and A. Newhouse (1958a), Applications of the monte carlo methods to architectural acoustics, *JASA*, *30*(1), 1–3. 34, 48

Allred, J. C., and A. Newhouse (1958b), Applications of the monte carlo methods to architectural acoustics ii, *JASA*, *30*(10), 903–904. 34, 48

Bang, and Olufsen (1992), Music for archimedes. 113

Bech, S. (1998), Spatial aspects of reproduced sound in small rooms, *JASA*, *103*(1), 434–445. 100

Begault, D. R., B. U. McClain, and M. R. Anderson (2001), Early reflection thresholds for virtual sound sources, in *Procedings of the International Workshop on spatial Media*, Aizu-Wakamatsu, Japan. 100, 109, 110, 114

Bekaert, P. (1999), Hierarchical and stochastic algorithms for radiosity, Ph.D. thesis, Catholieke Universiteit Leuven. 61

Beranek, L. (2003), *Concert Halls and Opera Houses: Music, Acoustics, and Architecture*, Springer-Verlag. 31

Bertram, M., E. Deines, J. Mohring, J. Jerorovs, and H. Hagen (2005), Phonon tracing for auralization and visualization of sound, in *Procedings of 16th IEEE Visualisation, 2005*, pp. 20–27. 35, 47

Blauert, J. (1969), Sound localization in the median plane, *Acta Acustica*, *22*, 205 – 213. 102, 103

Blauert, J. (1999), *Spatial Hearing, The Psychophysics of Human Sound Localisation*, MIT Press. viii, xi, 25, 27, 87, 100, 102, 104, 105, 109

Borish, J. (1984), Extension of the image model to arbitrary polyhedra, *JASA*, *75*(6), 1827–1836. 46, 47

Bork, I. (2005a), Report on the 3rd round robin on room acoustical computer simulation - part i : Measurements, *Acta Acustica*, *91*, 740–752. ix, 26, 31, 40, 111, 138, 139, 144, 157, 161

Bork, I. (2005b), Report on the 3rd round robin on room acoustical computer simulation - part ii : Calculation, *Acta Acustica*, *91*, 753–763. ix, xi, 4, 111, 137, 138, 139, 141, 144, 145, 146, 157, 160, 161

Bresenham, J. E. (1965), Algorithm for computer control of a digital plotter, *IBM Systems Journal*, *4*(1), 25–30. 9

Cadoz, C., A. Luciani, and J. L. Florens (1993), CORDIS-ANIMA. a modeling and simulation system for sound and image synthesis. the general formalism, *Computer Music J.*, *17*(1), 19–29. 6

Calamia, P. T. (2009), Advances in edge-diffraction modeling for virtual-acoustic simulations, Ph.D. thesis, Princeton University, june. 36

Cox, T., B.-I. L. Dalenback, P. D'Antonio, J. J. Embrechts, J. Y. Jeon, E. Mommertz, and M. Vorländer (2006), A tutorial on scattering and diffusion coefficients for room acoustic surfaces, *Acta Acustica*, *92*, 1–15. vii, 39

Crochiere, R. (1980), A weighted overlap-add method of short-time fourier analysis/synthesis, *IEEE Transactions on Acoustics, Speech and Signal Processing*, *28*(1), 99 – 102. 93

Dalenbäck, B.-I. L. (1996), Room acoustic prediction based on a unified treatment of diffuse and specular reflection, *Applied Acoustics*, *100*(2), 899–909. 47, 55

Dalenbäck, B.-I. L., M. Kleiner, and P. Svensson (1994), A macroscopic view of diffuse reflection, *JAES*, *42*(10), 793–805. 55

Daniel, J. (2001), Représentation de champs acoustiques, application à la transmission et à la représentation de scènes sonores complexes dans un contexte multimédia, Ph.D. thesis, Université Paris 6. 29

Deille, O., J. Maillard, N. Noé, K. Bouatouch, and J. Martin (2006a), Real time acoustic rendering of complex environments including diffraction and curved surfaces, in *Procedings of 120th Audio Enginneering Society Convetion Paper (AES)*, Paris, France. vii, 3, 4, 21, 27, 75, 76, 77, 78, 79, 81, 82, 120, 121

Deille, O., J. Maillard, N. Noé, K. Bouatouch, and J. Martin (2006b), Interactive real-time auralization system of complexly-shaped volumes, in *Procedings of Euronoise 2006*, Tampere, Finland. 75, 77

Deines, E., F. Michel, M. Bertram, H. Hagen, and G. M. Nielson (2006), Visualizing the phonon map, *Proceedings of Eurographics/ IEEE-VGTC Symposium on Visualization (2006)*, pp. 291–298. 36

Dutre, P., K. Bala, and P. Bekaert (2002), *Advanced Global Illumination*, A. K. Peters, Ltd., Natick, MA, USA. 12

Embrechts, J.-J. (2000), Broad spectrum diffusion model for room acoustics ray-tracing algorithms, *J.A.S.A*, *107*(4), 2068–2080. 53, 71

Emerit, M. (1995), Simulation binaurale de l'acoustique de salles de concert, Ph.D. thesis, Institut National Polytechnique de Grenoble. 25, 27

Emerit, M., E. Dudouet, and J. Martin (1995), Head related transfer functions and high-order statistics, in *Procedings of 15th International Congress On Acoustics, Trondheim, 1995*, pp. 437–440. 27, 77, 79, 82, 111

Foote, B., and J. Yoder (1998), Metadata and active object-models, in *in OOPSLA Metadata workshop*, Vancouver, Canada. 127, 131

Funkhouser, T., N. Tsingos, I. Carlbom, G. Elko, M. Sondhi, J. E. West, G. Pingali, P. Min, and A. Ngan (2004), A beam tracing method for interactive architectural acoustics, *JASA*, *115*(2), 739–756. 1, 47, 120, 121

Galletti, O. (2010), Création d'un ordonnanceur pour le rendu sonore dynamique d'environnements complexes, *Tech. rep.*, CSTB. 36

Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison and Wesley. 125

García, G. (2002), Optimal filter partition for efficient convolution with short input/output delay, in *Procedings of 113th Audio Engineering Society Convention Paper (AES)*, Los Angeles, California, USA. 94

Gardner, M. B. (1969), Distance estimation of 0 ° or apparent 0 ° oriented speech signals in anechoic space, *JASA*. 101

Gardner, W. G. (1995), Efficient convolution without input-output delay, *J. Audio Eng. Soc*, *43*(3), 127–136. 94

Govindaraju, N. K., B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli (2008), High performance discrete fourier transforms on graphics processors, in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pp. 1–12, IEEE Press, Piscataway, NJ, USA, doi:http://doi.acm.org/10.1145/1413370.1413373. 91

Guenther, R. B., and J. W. Lee (1996), *Partial differential Equations of Mathematical Physics and Integral Equations*, Dover Publications Inc. 15

Guicquero Le Beyec, W. (2010), Optimization of dynamic sound rendering algorithms for reverberated environment, *Tech. rep.*, CSTB. 94

Hacıhabiboğlu, H., and F. Murtagh (2006), An observational study of the precedence effect, *Acta Acustica*, *92*(3), 440–456. viii, 105, 109

Hacıhabiboğlu, H., and F. Murtagh (2008), Perceptual simplification for model-based binaural room auralisation, *Applied Acoustics*, *69*, 715–272. viii, 1, 3, 4, 99, 100, 106, 107, 108

Hanyu, T. (2010), A theoretical framework for quantitatively characterizing sound field diffusion based on scattering coefficient and absorption coefficient of walls, *JASA*, *128*(3), 1140 – 1148. 40

Hartmann, W. M. (1983), Localisation of sound in rooms, *JASA*, *74*(5), 1380–1391. 103, 104

Hartmann, W. M., and B. Rakerd (1989), Localisation of sound in rooms iv : The franssen effect, *JASA*, *86*(4), 1366–1373. 103

Heckbert, P. S. (1990), Adaptive radiosity textures for bidirectional ray tracing, in *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pp. 145–154, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/97879.97895. 55

Hermant, N. (2010), Limitations and improvements for a beam tracing simulation software applied to room acoustics, Master's thesis, Chalmers University of Technology. 44

Holtzchuch, N. (2009), Computer graphics ii: Rendering, Master of Science in Informatics at Grenoble – Course. 149, 150

ISO3382-1 (2009), Acoustics – measurement of room acoustic parameters – part 1: Performance spaces, *International Norm, ISO 3382-1:2009*. 157

ISO9613-3 (1996), Acoustics – attenuation of sound during propagation outdoors – part 1: Calculation of the absorption of sound by the atmosphere, *International Norm, ISO 9613-3:1996*. 6, 27

Jensen, H. W. (2001), *Realistic Image Synthesis Using Photon Mapping*, AK Peters, Ltd. 48, 55

Jot, J.-M. (1992), étude et réalisation d'un spatialisateur de sons par modèles physique et perceptifs, Ph.D. thesis, ENST. 87

Jouhaneau, J. (2000), *Notions élémentaires d'acoustique - Electroacoustique*, Acoustique Appliqué Collection. 30

Kahrs, M., and K. Brandenburg (1998), *Applications of digital signal processing to audio and acoustics*, Kluwer Academic Publishers, Norwell, MA, USA. 17, 22, 34

Kajastila, R., T. Lokki, P. Lundén, L. Savioja, and S. Siltanen (2007), A distributed real-time virtual acoustic rendering system for dynamic geometries, in *Procedings of 122th Audio Enginneering Society Convetion Paper (AES)*, Vienna, Ausrtia. 120

Kajiya, J. T. (1986), The rendering equation, *SIGGRAPH Comput. Graph.*, *20*, 143–150, doi:http://doi.acm.org/10.1145/15886.15902. 12, 13, 15, 48

Kapralos, B. (2006), The sonel mapping acoustical modeling methods, Ph.D. thesis, York University Toronto, Ontario. 37, 47, 55

Kapralos, B., M. Jenkin, and E. Milios (2005), Acoustical modeling using a russian roulette strategy, in *Procedings of 118th Audio Engineering Society Convention Paper (AES)*, Barcelona, Spain. 52

Kapralos, B., M. Jenkin, and E. Milios (2006), Sonel mapping : A stochastic acoustical modeling system, in *Proecdings of ICASSP 2006 : IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 421–424. 35

Kapralos, B., M. Jenkin, and E. Milios (2007), Diffraction modeling for interactive virtual acoustical environments, in *Proceedings of the 2nd International Conference on Computer Graphics Theory and Applications (GRAPP)*, Barcelona, Spain. 120

Katz, B. F. G. (2004), International round robin on room acoustical impulse response analysis software, *ARLO - Acoustics Research Letters Online*, *5*(4), 158 – 164. 139

Keller, J. B. (1961), Geometrical theory of diffraction, *Journal of the Optical Society of America*, *2*(2), 116–130. 36, 121, 148

Kiminki, S. (2005), Sound propagation theory for linear ray acoustic modeling, Master's thesis, Helsinki University of Technology. 6, 12, 13, 15

Kleiner, M., B.-I. L. Dalenbäck, and P. Svensson (1993), Auralization - an overview, *JAES*, *41*(11), 861–875. 79

Krokstad, A., S. Strom, and S. Sorsdal (1968), Calculating the acoustical room impulse response by the use of a ray-tracing technique, *Journal of Sound and Vibrations*, *8*, 118. 41

Kulowski, A. (1985), Algorithmic representation of the ray tracing technique, *Applied Acoustics*, *18*(6), 449–469. 34, 35, 41

Kuttruff, H. (1973), *Room Acoustics*, Applied Science Publishers. 29, 31, 40

Kuttruff, K. H. (1993), Auralization of impulse responses modeled on the basis of ray-tracing results, *J. Audio Eng. Soc*, *41*(11), 876–880. 78, 85

Lafortune, E. P., and Y. D. Willems (1993), Bi-directional path tracing, in *PROCEEDINGS OF THIRD INTERNATIONAL CONFERENCE ON COMPUTATIONAL GRAPHICS AND VISUALIZATION TECHNIQUES (COMPUGRAPHICS '93*, pp. 145–153. 48

Lauterbach, C., A. Chandak, and D. Manocha (2007), Interactive sound rendering in complex and dynamic scenes using frustum tracing, *IEEE Transactions on Visualization and Computer Graphics*, *13*, 1672–1679, doi:http://dx.doi.org/10.1109/TVCG.2007.70567. 120

Lentz, T., D. Schröder, M. Vorländer, and I. Assenmacher (2007), Virtual reality system with integrated sound field simulation and reproduction, *EURASIP : Journal on Advances in Signal Processing*. vii, 42, 43, 120

Lesoinne, S. (2006), Techniques d'accélération du tir de rayons pour la simulation acoustique, Master's thesis, Université de Liège, Faculté de Sciences Appliquées. 43

Lesoinne, S., and J.-J. Embrechts (2008), Size-adaptative spherical receptor acceleration method for acoustical ray tracing, in *Acoustics'08*. 43, 44

Lesoinne, S., N. Werner, and J. Embrechts (2006), 3d real-time auralization with separate rendering of direct sound, reflections and directional late reverberation, in *Proceedings of the 12th International Conference on Auditory Display*, London, UK. 120

Litovsky, R. Y., S. H. Colburn, W. A. Yost, and S. J. Guzman (1999), The precedence effect, *JASA*, *106*(4), 1633–1654, doi:http://dx.doi.org/10.1121/1.427914. viii, 104, 105, 109

Loyet, R. (2007), Optimisation du rendu sonore d'environnements réverbérés, Master's thesis, INP Grenoble. 124, 127

Loyet, R., J. Maillard, J.-C. Iehl, and B. Péroche (2009), Perceptual clustering for ray based auralization, in *Procedings of Euronoise 2009*, Edinburgh, Scotland. 111

Maillard, J. (2009), Prediction and auralization of construction site noise, in *Procedings of Euronoise 2009*, Edinburgh, Scotland. 84

Mercier, D., G. Kisselhoff, J.-L. Ohl, and P. Durovic (2006), *Le livre des techniques du son : Tome 3. L'exploitation*, Dunod. 10

Millet, A. (), Méthodes de monte-carlo, Master 2ème année : Spécialité Modélisation Aléatoire – Course. 155

Morfey, C. (2000), *The Dictionary of Acoustics*, Academic Press. 7

Müller, S., and P. Massarani (2001), Transfer-function measurement with sweeps, *JAES*, pp. 443–471. 19, 29

Munshi, A. (2009), The opencl specification, *Tech. rep.*, Khronos Group. 89, 120

Nicol, R., J. Daniel, M. Emerit, G. Pallone, D. Virette, N. Chetry, P. Guillon, and S. Bertet (2008), Le son 3d dans toutes ses dimensions, *Acoustique et Technique*, *52*, 43–50. 25

Niemitalo, O. (2001), Polynomial interpolators for high-quality resampling of oversampled audio, unpublished. viii, 151, 152, 153

Noisternig, M., B. F. Katz, S. Siltanen, and L. Savioja (2008), Framework for real-time auralization in architectural acoustics, *Acta Acustica united with Acustica*, *94*, 1000–1015(16), doi:doi:10.3813/AAA.918116. 120

NVidia (2009), Nvidia opencl best practices guide, *Tech. rep.*, NVidia. 93

Oppenheim, A. V., and R. W. Schafer (1975), *Digital signal processing*, Prentice Hall. 21, 79

Peter, S. U., F. I. Roger, and V. John (1999), An analytic secondary source model of edge diffraction impulse responses, *JASA*, *106*(5), 2331–2344. 36

Pharr, M., and G. Humphreys (2004), *Physically Based Rendering: From Theory to Implementation*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. vi, 37, 52, 150, 151, 154, 155

Preibisch-Effenberger, R. (1966), Die schallokalisationsfähigkeit des menschen und ihre audiometrische verwendung zut klinischen diagnostik [the human faculty of sound localization and its audiometric application to clinical diagnostics], Ph.D. thesis, Dresden Technische Universität. 102

Pulkki, V. (1997), Virtual sound source positionong using vector base amplitude panning, *JAES*, *45*(6), 456–466. 28

Pulkki, V., M. Karjalainen, and J. Huopaniemi (1999), Analyzing virtual sound source attributes using a binaural auditory model, *JAES*, *47*(4), 203–217. 28

Rakerd, B., and W. M. Hartmann (1985), Localisation of sound in rooms ii : The effects of a single reflecting surface, *JASA*, *78*(2), 524–533. 103, 104

Rakerd, B., and W. M. Hartmann (1986), Localisation of sound in rooms iii : Onset and duration effects, *JASA*, *80*(6), 1695–1706. 103, 104

Röber, N., U. Kaminski, and M. Masuch (2007), Ray acoustics using computer graphics technology, in *Procedings of 10th International Conference on Digital Audio Effects (DAFx)*, Bordeau, France. 42, 43, 120

Savioja, L., H. Jyri, Lokki, and Väänänen (1999), Creating interactive virtual acoustic environments, *JAES*, *47*(9), 675–705. 79, 83

Schwark, M., U. Reiter, and A. Dantele (2004), Audiovisual virtual environments: Enabling realtime rendering of early reflections by scene graph simplification, in *Audio Engineering Society Convention 116*. 120

Segovia, B. (2007), Interactive light transport with virtual point lights, Ph.D. thesis, Université Claude Bernard - Lyon I. 138

Sibbald, A. (2009), Transaural acoustic crosstalk cancellation, *Tech. rep.*, Sensaura. 28

Siltanen, S., T. Lokki, S. Kiminki, and L. Savioja (2007), The room acoustic rendering equation, *JASA*, *122*(3), 1624–1635. 6, 12, 13, 14, 16, 47, 122

Siltanen, S., T. Lokki, L. Savioja, and C. L. Christensen (2008), Geometry reduction in room acoustics modeling, *Acta Acustica united with Acustica*, *94*, 410–418(9), doi:doi:10.3813/AAA.918049. 9

Siltanen, S., T. Lokki, and S. Lauri (2009), Frequency domain acoustic radiance transfer for real-time auralization, *Acta Acustica*, *95*(1), 110–117. viii, 1, 89, 120, 122, 123

Smith, J. O. (2009), *Spectral Audio Signal Processing*, W3K Publishing. 22, 24, 90, 93

Smith, S. W. (1997), *The scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing. 17, 18, 19, 21, 90

Soize, C. (1993), *Méthodes mathématiques en analyse du signal*, Masson. 17

Soo, J.-S., and K. K. Pang (1990), Multidelay block frequency domain adaptive filter, in *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. 38 - 2, pp. 373–376. 94

Stavrakis, E., N. Tsingos, and P. Calamia (2008), Topological sound propagation with reverberation graphs, *Acta Acustica*, *94*(6), 921–932. 59, 83

Stephenson, U. (1990), Comparison of the mirror image source method and the sound particle simulation method, *Applied Acoustics*, *29*(1), 35 – 72, doi:10.1016/0003-682X(90)90070-B. 35

Stevens, S. S., and E. B. Newman (1936), The localization of actual sources of sound, *AM. J. Psychol.*, *48*, 297 – 306. 101, 102

Stroustrup, B. (1997), *C++ Programming Language*, 3rd ed., Addison-Wesley. 120, 131

Svensson, U. P. (2002), Modeling acoustic spaces for audio virtual reality, in *Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002)*, Leuven, Belgium. 120

Tanguy, J.-P. (2007), *Traitement du signal — Théorie et pratique du signal — Signaux déterministes et aléatoires en continu et en discret*, Ellipses. 17

Tisserand, E., J.-F. Pautex, and P. Schweitzer (2008), *Analyse et traitement des signaux — Méthodes et applications au son et à l'image*, 2 ed., Dunod. 17

Tsingos, N. (1998), Simulation de champs sonores de haute qualité pour des applications graphiques interactives, Ph.D. thesis, Université Joseph Fourier-Grenoble 1. 25, 26, 37, 49

Tsingos, N., T. Funkhouser, N. Addy, and I. Carlbom (2000), Geometrical theory of diffraction for modeling acoustics in virtual environments. 37

Tsingos, N., C. Dachsbacher, S. Lefebvre, and M. Dellepiane (2007), Extending geometrical acoustics to highly detailed architectural environments, in *Procedings of ICA 2007 : 19th International Congress on Acoustics*, Madrid, Spain. 39

Tsuchiyama, R., T. Nakamura, T. Iizuka, A. Asahara, and S. Miki (2010), *The OpenCL Programming Book*, Fixstars Corporation. 93

van Maercke, D., and J. Martin (1993), The prediction of echograms and impulse responses within the epidaure software, *Applied Acoustics*, *38*(2-4), 93 – 114, doi:DOI:10.1016/0003-682X(93)90045-8. 47, 79

Veach, E. (1997), Robust monte carlo methods for light transport simulation, Ph.D. thesis, Department of Computer Science, Stanford University. 55

Veach, E., and L. J. Guibas (), Metropolis light transport, in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. 48

Vorländer, M. (1989), Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm, *JASA*, *86*(1), 172–178. 41, 42

Vorländer, M. (2008), *Auralization — Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality*, Springer. 10, 12, 25, 31, 34, 35, 40, 79

Wald, I., and V. Havran (2006), In building fast kd-trees for ray tracing, and on doing that in $o(nlogn)$, *Tech. rep.*, University of Utah SCI Institute (UUSCI). 138

Wallach, H., E. B. Newmaw, and M. R. Rosenzweig (1949), The precedence effect in sound localization, *Am. J. Psychol.*, *LXII*(3), 315 – 336. 104

Wong, T.-T., W.-S. Luk, and P.-A. Heng (1997), Sampling with hammersley and halton points, *J. Graph. Tools*, *2*, 9–24. 36