Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition

Nikunj Raghuvanshi, Rahul Narain and Ming C. Lin

Department of Computer Science, UNC Chapel Hill, USA

Abstract—Accurate sound rendering can add significant realism to complement visual display in interactive applications, as well as facilitate acoustic predictions for many engineering applications, like accurate acoustic analysis for architectural design [27]. Numerical simulation can provide this realism most naturally by modeling the underlying physics of wave propagation. However, wave simulation has traditionally posed a tough computational challenge. In this paper, we present a technique which relies on an adaptive rectangular decomposition of 3D scenes to enable efficient and accurate simulation of sound propagation in complex virtual environments. It exploits the known analytical solution of the Wave Equation in rectangular domains, and utilizes efficient implementation of Discrete Cosine Transform on the GPU to achieve at least a hundred-fold performance gain compared to a standard Finite Difference Time Domain (FDTD) implementation with comparable accuracy, while also being an order of magnitude more memory-efficient. Consequently, we are able to perform accurate numerical acoustic simulation on large, complex scenes in the kilohertz range. To the best of our knowledge, it was not previously possible to perform such simulations on a desktop computer. Our work thus enables acoustic analysis on large scenes and auditory display for complex virtual environments on commodity hardware.

1 Introduction

Sound rendering, or auditory display, was first introduced to computer graphics more than a decade ago by Takala and Hahn [44], who investigated the integration of sound with interactive graphics applications. Their work was motivated by the observation that accurate auditory display can augment graphical rendering and enhance human-computer interaction. There have been studies showing that such systems provide the user with an enhanced spatial sense of presence [14]. Auditory display typically consists of two main components: *sound synthesis* that deals with how sound is produced [6, 13, 29, 33, 49] and *sound propagation* that is concerned with how sound travels in a scene. In this paper we address the problem of sound propagation, also referred to as computational acoustics.

The input to an acoustic simulator is the geometry of the scene, along with the reflective properties of different parts of the boundary and the locations of the sound sources and listener. The goal is to *auralize* – predict the sound the listener would hear. Computational acoustics has a very diverse range of applications, from noise control and underwater acoustics [21] to architectural acoustics and acoustics for virtual environments (VEs) and games. Although each application has its own unique requirements from the simulation technique, all applications require physical accuracy. For noise control, accuracy translates directly into the loudness of the perceived noise, for architectural acoustics, accuracy has implications on predicting how much an orchestra theater enhances (or degrades) the quality of music. For interactive applications like VEs and games, physical accuracy directly affects the perceived realism and immersion of the scene. This is because we are used to observing many physical wave effects in reality and their presence in the scene helps to convince us that the computer-generated environment is indeed real. For example, we observe every day that when a sound source is occluded from the listener, the sound becomes "muffled" in reality. For light, the source would become invisible, casting a shadow, which doesn't happen for sound because it bends, or *diffracts*, around the occluder. In fact, this is one of the major reasons that sound compliments sight, both in reality and in virtual environments – it conveys information in places where light cannot. Our simulation technique naturally captures these subtle phenomena occurring in nature.

For most acoustic simulation techniques, the process of auralization can be further broken down into roughly two parts: (a) pre-processing;

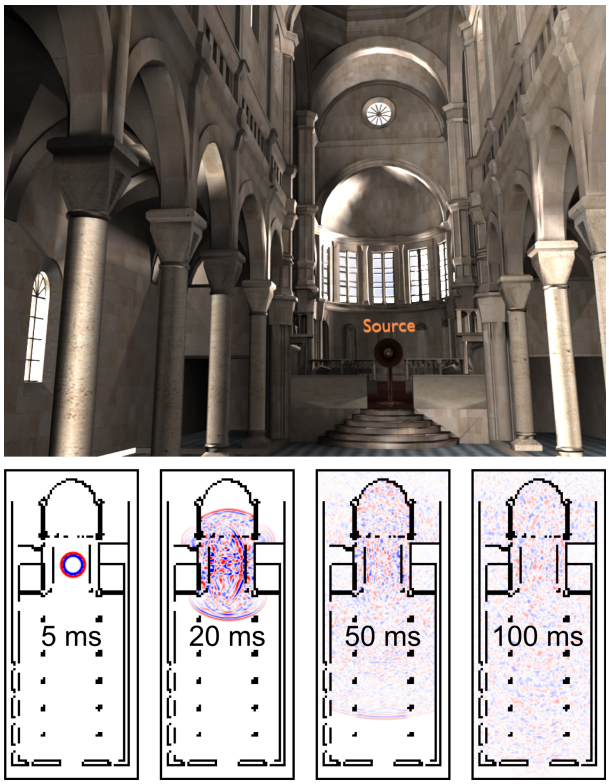


Fig. 1. Sound simulation on a Cathedral. The dimensions of this scene are $35m \times 15m \times 26m$. We are able to perform numerical sound simulation on this complex scene on a desktop computer and pre-compute a 1 second long *impulse response* in about 29 minutes, taking less than 1 GB of memory. A commonly used approach that we compare against, Finite Difference Time Domain (FDTD), would take 1 week of computation and 25GB of memory for this scene to achieve competitive accuracy. The auralization, or sound rendering at runtime consists of convolution of the calculated impulse responses with arbitrary source signals, that can be computed efficiently.

and (b) (acoustic) sound rendering. During pre-processing, an acoustic simulator does computations on the environment to estimate its acoustical properties, which facilitate fast rendering of the sound at runtime. The exact pre-computation depends on the specific approach being used. For our approach, the pre-processing consists of running a simulation from the source location, which yields the impulse responses at *all* points in the scene in one simulation. The rendering at runtime can then be performed by convolving the source signal with the calculated impulse response at the listener's location, which is a very fast operation as it can be performed through an FFT. The main focus of this paper is on the pre-processing phase of acoustic prediction. We present a novel and fast numerical approach that enables efficient and accurate acoustic simulations on large scenes on a desktop system in minutes, which would have otherwise taken many days of computation on a small cluster. An example is shown in Figure 1.

The problem of acoustic simulation is challenging mainly due to some specific properties of sound. The wavelength of audible sound falls exactly in the range of the dimensions of common objects, in the order of a few centimeters to a few meters. Consequently, unlike light, sound bends (*diffracts*) appreciably around most objects, especially at lower frequencies. This means that unlike light, sound doesn't exhibit any abrupt shadow edges. As discussed earlier, from a perceptual point of view, capturing correct sound diffraction is critical. In addition, the speed of sound is small enough that the temporal sequence of multiple sound reflections in a room is easily perceptible and distinguishable by humans. As a result, a steady state solution, like in light simulation, is insufficient for sound – a full transient solution is required. For example, speech intelligibility is greatly reduced in a room with very high wall reflectivity, since all the echoes mix into the direct sound with varying delays. Therefore, the combination of low speed and large wavelength makes sound simulation a computational problem with its own unique challenges. Numerical approaches for sound propagation attempt to directly solve the Acoustic Wave Equation, which governs all linear sound propagation phenomena, and are thus capable of performing a full transient solution which correctly accounts for all wave phenomena, including diffraction, elegantly in one framework. Since we use a numerical approach, our implementation inherits all these advantages. This is also the chief benefit our method offers over geometric techniques, which we will discuss in detail in the next section.

Applicability: Most interactive applications today, such as games, use reverb filters (or equivalently, impulse responses) that are not physically-based and roughly correspond to acoustical spaces with different sizes. In reality, the acoustics of a space exhibits perceptibly large variations depending on the wall material, room size and geometry, along with many other factors [24]. A handful of reverb filters common to all scenes cannot possibly capture all the different acoustical effects which we routinely observe in real life and thus, such a method at best provides a crude approximation of the actual acoustics of the scene. Moreover, an artist has to assign these reverb filters to different parts of the environment manually, which requires a considerable amount of time and effort.

One way to obtain realistic filters would be to do actual measurements on a scene. Not only is it difficult and time-consuming for real scenes, but for virtual environments and games, one would need to physically construct scale physical prototypes which would be prohibitively expensive. This is even more impractical considering that most games today encourage users to author their own scenes. Numerical approaches offer a cheap and effective alternative to alleviate all of these problems by computing the filters at different points in the scene directly from simulation and are thus capable of at once automating the procedure, as well as providing much more realistic and immersive acoustics which account for all perceptually-important auditory effects, including diffraction. However, this realism comes at a very high computational cost and large memory requirements. In this paper, we offer a highly accelerated numerical technique that works on a desktop system and can be used to pre-compute high-quality reverb filters for arbitrary scenes without any human intervention. These filters can then be employed as-is in interactive applications for real-time auralization. For example, given that the artist specifies a few salient locations where the acoustics must be captured, one just needs to store the reverb filters obtained from simulation at those locations. Current game engines already use techniques to associate reverb filters with physical locations [2]. Our technique would provide the actual values in the reverb filters, the audio pipeline need not change at all. The artist would thus be relieved from the burden of figuring out and experimenting exactly what kind of reverb captures the acoustics of the particular space he/she has modeled. Another advantage of our approach is that since we are solving for the complete sound field in a scene, a sound designer can visualize how the sound propagates in the scene over time, to help him/her make guided decisions about what changes need to be made to the scene to counter any perceived acoustic deficiencies. Please refer to the accompanying video for examples of such visualizations.

Main Results: Our technique takes at least an order of magnitude less memory and two orders of magnitude less computation compared to a standard numerical implementation, while achieving competitive accuracy at the same time. It relies on an *adaptive rectangular decomposition* of the free space of the scene. This approach has many advantages:

1. The analytical solution to the wave equation within a rectangular domain is known. This enables high numerical accuracy, even on grids approaching the Nyquist limit, that are much coarser than those required by most numerical techniques. Exploiting these analytical solutions is one of the key reasons for the significant reduction in compute and memory requirements.

2. Owing to the rectangular shape of the domain partitions, the solution in their interior can be expressed in terms of the Discrete Cosine Transform (DCT). It is well-known that DCTs can be efficiently calculated through an FFT. We use a fast implementation of FFT on the GPU [17], that effectively maps the FFT to the highly parallel architecture of the GPU to gain considerable speedups over CPU-based libraries. This implementation drastically reduces the computation time for our overall approach.

3. The rectangular decomposition can be seamlessly coupled with other simulation techniques running in different parts of the simulation domain.

We have also implemented the Perfectly Matched Layer (PML) Absorber to model partially absorbing surfaces, as well as open scenes. We demonstrate our algorithm on several scenarios with high complexity and validate the results against FDTD, a standard Finite Difference technique. We show that our approach is able to achieve the same level of accuracy with at least two orders of magnitude reduction in computation time and an order of magnitude less memory requirements. Consequently, we are able to perform accurate numerical acoustic simulation on large scenes in the kilohertz range which, to the best of our knowledge, have not been previously possible on a desktop computer.

Organization: The rest of the paper is organized as follows. In Section 2, we review related work in the field. Section 3 presents the mathematical background, which motivates our approach described in Section 4. We show and discuss our results in Section 5.

2 Previous Work

Since its inception [36], computational acoustics has been a very active area of research due to its widespread practical applications. Depending on how wave propagation is approximated, techniques for simulating acoustics may be broadly classified into Geometric Acoustics (GA) and Numerical Acoustics (NA). For a general introduction to room acoustics, the reader may refer to [21, 24] or a more current survey [26].

Geometric Acoustics: All GA approaches are based on the basic assumption of rectilinear propagation of sound waves, just like light.

Historically, the first GA approaches that were investigated were Ray Tracing and the Image Source Method [3, 23]. Most room acoustics software use a combination of these techniques to this day [35]. Another efficient geometric approach that has been proposed in literature, with emphasis on interactive graphics applications, is Beam Tracing [4, 16]. On the lines of Photon Mapping, there has been work on Phonon Tracing [5, 12] in acoustics. Also, researchers have proposed applying hierarchical radiosity to acoustical energy transfer [18, 46]. All GA approaches assume that sound propagates rectilinearly in rays, which results in unphysical sharp shadows and some techniques must be applied to ameliorate the resulting artifacts and include diffraction into the simulation, especially at lower frequencies. Most of such approaches rely on the Geometrical Theory of Diffraction [48] and more recently, the Biot-Tolstoy-Medwin model of diffraction [10] which result in improved simulations. However, accurately capturing diffraction still remains a challenge for GA approaches and is an active area of research. In the context of interactive systems, most acoustic techniques explored to date are based on GA, simply because although numerical approaches typically achieve better quality results, the computational demands were out of the reach of most systems.

Numerical Acoustics: Numerical approaches, in contrast to GA, solve the Wave Equation numerically to obtain the exact behavior of wave propagation in a domain. Based on how the spatial discretization is performed, numerical approaches for acoustics may be roughly classified into: Finite Element Method (FEM), Boundary Element Method (BEM), Digital Waveguide Mesh (DWM), Finite Difference Time Domain (FDTD) and Functional Transform Method (FTM). In the following, we briefly review each of these methods in turn.

FEM and BEM have traditionally been employed mainly for the steady-state frequency domain response, as opposed to a full time-domain solution, with FEM applied mainly to interior and BEM to exterior scattering problems [22]. FEM and BEM approaches are general methods applicable to any Partial Differential Equation, the Wave Equation being one of them. DWM approaches [50], on the other hand, use discrete waveguide elements, each of which is assumed to carry waves along its length along a single dimension [20, 28, 40].

The FDTD method, owing to its simplicity and versatility, has been an active area of research in room acoustics for more than a decade [7, 8]. Originally proposed for electromagnetic simulations [41], FDTD works on a uniform grid and solves for the field values at each cell over time. Initial investigations into FDTD were hampered by the lack of computational power and memory, limiting its application to mostly small scenes in 2D. It is only recently that the possibility of applying FDTD to medium sized scenes in 3D has been explored [37–39]. Even then, the computational and memory requirements for FDTD are beyond the capability of most desktop systems today [39], requiring days of computation on a small cluster for medium-sized 3D scenes for simulating frequencies up to 1 kHz.

Another aspect of our work is that we divide the domain into many partitions. Such approaches, called Domain Decomposition Methods (DDM) have widespread applications in all areas where numerical solution to partial differential equations is required and it would be hard to list all areas of numerical simulation where they have been applied. For a brief history of DDM and its applications we refer the reader to the survey [11]. For an in-depth discussion of DDM, the reader is referred to the books [31, 45]. Also, the website [1] has many references to current work in the area. It is interesting to note that the main motivation of DDM when it was conceptualized more than a century ago was to divide the domain into simpler partitions which could be analyzed more easily [11], much like in our work. However, in nearly all Domain Decomposition approaches today, specifically for wave propagation, the principal goal is to divide and parallelize the workload across multiple processors. Therefore, the chief requirement in such cases is that the partitions be of equal size and have minimal interface area, since that corresponds to balancing the computation and minimizing the communication cost.

The motivation of our approach for partitioning the domain is different – we want to ensure that the partitions have a particular *rectangular shape* even if that implies partitions with highly varying sizes since it yields many algorithmic improvements in terms of computation and numerical accuracy for simulation within the partitions. Our approach leads to improvements even in sequential performance by exploiting the analytical solution within a rectangular domain. In contrast to prior work in high-performance computing, parallelization is not the driving priority in our work. Decomposing the domain into partitions and performing interface handling between them are very well-known techniques and by themselves are not the main contribution of this work. Of course, it is still possible to parallelize our approach by allocating the partitions to different cores or machines, and doing interface handling between them, and would be the way to scale our approach to very large scenes with billions of cells.

Another method which is related to our work, although in a different mathematical framework, is the Functional Transform Method (FTM) [30, 32]. Our technique has the advantage of being very simple to formulate and works directly on the second order Wave Equation, instead of casting it as a first order system as in the FTM and just requires one mathematical transformation, the Discrete Cosine Transform. Also, we demonstrate our results on general scenes in 3D, along with detailed error analysis.

Spectral techniques are a class of very high order numerical schemes in which the complete field is expressed in terms of global basis functions. Typically, the basis set is chosen to be the Fourier or Chebyshev polynomials [9] as fast, memory efficient transforms are available to transform to these bases from physical space and vice versa. Our method may also be regarded as a spectral method. However, there are some important differences which we discuss later in the paper.

It is interesting to note here that GA and NA approaches may be regarded as complimentary with regard to the range of frequencies they can simulate efficiently – With geometric approaches it is hard to simulate low-frequency wave phenomena like diffraction because they assume that sound travels in straight lines like light, while with numerical approaches, simulating high frequencies above a few kilohertz becomes prohibitive due to the excessively fine volume mesh that must be created.

We wish to emphasize at this point that it is possible to integrate more elaborate techniques for modeling the surface properties and scattering [47] characteristics of the scene boundary into our technique. Also, we assume all sound sources to be monopole, or point source, but complex emission patterns resulting from many monopole and dipole sources [19] can also be easily integrated in our framework.

3 Mathematical Background

In this section, we first briefly present the FDTD method. We do this for two reasons: Firstly, this is the simulator we use as a reference to compare against and its details serve to illustrate the underlying mathematical framework used throughout this paper. Secondly, this discussion illustrates numerical dispersion errors in FDTD and motivates our technique which uses the analytical solution to the Wave Equation on rectangular domains to remove numerical dispersion errors.

3.1 Basic Formulation

The input to an acoustics simulator is a scene in 3D, along with the boundary conditions and the locations of the sound sources and listener. The propagation of sound in a domain is governed by the Acoustic Wave Equation,

$$\frac{\partial^2 p}{\partial t^2} - c^2 \nabla^2 p = F(\mathbf{x},t), \tag{1}$$

This equation captures the complete wave nature of sound, which is treated as a time-varying pressure field $p(\mathbf{x},t)$ in space. The speed of sound is $c = 340ms^{-1}$ and $F(\mathbf{x},t)$ is the forcing term corresponding to sound sources present in the scene. The operator $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian in 3D. The Wave Equation succinctly explains wave phenomena such as interference and diffraction that are observed in reality. We briefly mention a few physical quantities and their relations, which will be used throughout the paper. For a wave traveling in

free space, the frequency, ν and wavelength, λ are related by $c = \nu\lambda$. It is also common to use the angular counterparts of these quantities: angular frequency, $\omega = 2\pi\nu$ and wavenumber, $k = \frac{2\pi}{\lambda}$. Because frequency and wavenumber are directly proportional to each other, we will be using the two terms interchangeably throughout the paper.

In the next sub-section, we briefly discuss the Finite Difference Time Domain (FDTD) method for numerically solving the Wave Equation. To avoid confusion, we note here that while the term FDTD is sometimes used to specifically refer to the original algorithm proposed by Yee [51] for Electromagnetic simulation, it is common to refer to any Finite Difference-based approach which computes the complete sound field in time domain as an FDTD method. In this paper, we use the latter definition.

3.2 A (2,6) FDTD Scheme

FDTD works on a uniform grid with spacing h. To capture the propagation of a prescribed maximum frequency ν_{max}, the Nyquist theorem requires that $h \leq \frac{\lambda_{max}}{2} = \frac{c}{2\nu_{max}}$. Once the spatial discretization is performed, the continuum Laplacian operator is replaced with a discrete approximation of desired order of accuracy. Throughout this paper, we consider the sixth order accurate approximation to the Laplacian, which approximates the second order differential in each dimension with the following stencil:

$$\frac{d^2p_i}{dx^2} \approx \frac{1}{180h^2}(2p_{i-3} - 27p_{i-2} + 270p_{i-1} - 490p_i + 270p_{i+1} - 27p_{i+2} + 2p_{i+3}) + O(h^6), \quad (2)$$

where p_i is the i^{th} grid cell in the corresponding dimension. Thus, the Laplacian operator at each cell can be represented as a Discrete Laplacian Matrix, K and equation (1) becomes,

$$\frac{\partial^2 P}{\partial t^2} - \frac{c^2}{h^2}KP = F(t), \quad (3)$$

where P is a long vector listing the pressure values at all the grid cells and F is the forcing term at each cell. Hard-walls may be modeled with the Neumann Boundary Condition – $\frac{\partial p}{\partial \hat{n}} = 0$, where $\hat{n}$ is the normal to the boundary.

The next step is to discretize equation (3) in time at some time-step Δt, which is restricted by the CFL condition $\Delta t < \frac{h}{c\sqrt{3}}$. Using the Leapfrog integrator in time, the complete update rule is as follows.

$$P^{n+1} = 2P^n - P^{n-1} + \left(\frac{c\Delta t}{h}\right)^2 KP^n + O\left(\Delta t^2\right) + O\left(h^6\right).$$

Since the temporal and spatial errors are second and sixth order respectively, this is a (2,6) FDTD scheme. In the next sub-section, we discuss the nature of the numerical errors in FDTD schemes and the resulting performance issues.

3.3 Numerical Dispersion in FDTD and Efficiency Considerations

As was previously discussed, the spatial cell size, h for FDTD is chosen depending on the maximum simulated frequency, ν_{max} and is limited by the Nyquist sampling theorem. However, due to numerical errors arising from spatial and temporal discretization, accurate simulation with FDTD typically requires not 2 but 8-10 samples per wavelength [43]. These errors manifest themselves in the form of Numerical Dispersion – Waves with different wavenumbers (or equivalently, different frequencies) do not travel with the same speed in the simulation. This error may be quantified by finding the wavenumber-dependent numerical speed, $c'(k)$, where k is the wavenumber. This speed is then normalized by dividing with the ideal wave speed, c, yielding the dispersion coefficient, $\gamma(k)$. Ideally, the dispersion coefficient should be as close to 1 as possible, for all wavenumbers. Figure 2 shows a plot of the dispersion coefficient for FDTD against frequency on a 3D grid and compares the error for different cell sizes. Observe

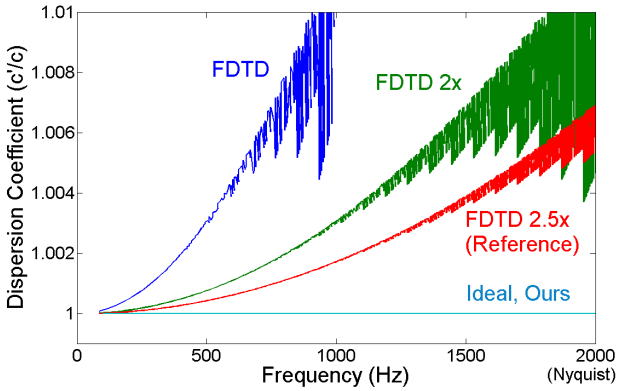


Fig. 2. Numerical dispersion with a (2,6) FDTD scheme for different mesh resolutions. Increasing the sampling reduces the numerical dispersion errors. Our method suffers from no dispersion errors in the interior of rectangular partitions, while FDTD accumulates errors over each cell a signal propagates across. Reducing these errors with FDTD requires a very fine grid.

that at 1000 Hz, the dispersion coefficient for FDTD is about $.01c$, while for FDTD running on a 2.5× refined mesh the error is about $.001c$. This is because the spatial sampling increases from 4 samples per wavelength to 10 samples per wavelength.

Consider a short-time signal containing many frequencies, for example, a spoken consonant. Due to Numerical Dispersion, each of the frequencies in the consonant will travel with a slightly different speed. As soon as the phase relations between different frequencies are lost, the signal is effectively destroyed and the result is a muffled sound. From the above values of the dispersion coefficient, it can be shown that with FDTD a signal would have lost phase coherence after traveling just 17m, which is comparable to the diameter of most scenes.

To increase accuracy, we need to increase the mesh resolution, but that greatly increases the compute and memory requirements of FDTD – Refining the mesh r times implies an increase on memory by a factor of r^3 and the total compute time for a given interval of time by r^4. In practice, memory can be a much tighter constraint because if the method runs out of main memory, it will effectively fail to produce *any* results.

3.4 Wave Equation on a Rectangular Domain

A lot of work has been done in the field of Spectral/Pseudo-spectral methods [25] to allow for accurate simulations with 2-4 samples per wavelength while still allowing for accurate simulations. Such methods typically represent the whole field in terms of global basis functions, as opposed to local basis functions used in Finite Difference or Finite Element methods. With a suitable choice of the spectral basis (typically Chebyshev polynomials), the differentiation represented by the Laplacian operator can be approximated to a very high degree of accuracy, leading to very accurate simulations. However, spectral methods still use discrete integration in time which introduces temporal numerical errors. In this paper, we use a different approach and instead exploit the well-known analytical solution to the Wave Equation on rectangular domains [24], which enables error-free propagation within the domain. It is important to note here that we are able to do this because we assume that the speed of sound is constant in the medium, which is a reasonable assumption for architectural acoustics and virtual environments.

Consider a rectangular domain in 3D, with its solid diagonal extending from the $(0,0,0)$ to (l_x, l_y, l_z), with perfectly rigid, reflective walls. It can be shown that any pressure field $p(x,y,z,t)$ in this domain may be represented as

$$p(x,y,z,t) = \sum_{i=(i_x,i_y,i_z)} m_i(t)\Phi_i(x,y,z), \quad (4)$$

where m_i are the time-varying mode coefficients and Φ_i are the eigenfunctions of the Laplacian for a rectangular domain, given by –

$$\Phi_i(x,y,z) = \cos\left(\frac{\pi i_x}{l_x}x\right)\cos\left(\frac{\pi i_y}{l_y}y\right)\cos\left(\frac{\pi i_z}{l_z}z\right).$$

Given that we want to simulate signals band-limited up to a prescribed smallest wavelength, the above continuum relation may be interpreted on a discrete uniform grid with the highest wavenumber eigenfunctions being spatially sampled at the Nyquist rate. Note that as long as the simulated signal is properly band-limited and all the modes are used in the calculation, this discretization introduces no numerical errors. This is the reason it becomes possible to have very coarse grids with only 2-4 samples per wavelength and still do accurate wave propagation simulations. In the discrete interpretation, equation (4) is simply an inverse Discrete Cosine Transform (iDCT) in 3D, with Φ_i being the Cosine basis vectors for the given dimensions. Therefore, we may efficiently transform from mode coefficients (M) to pressure values (P) as –

$$P(t) = iDCT(M(t)). \tag{5}$$

This is the main advantage of choosing a rectangular shape – because the eigenfunctions of a rectangle are Cosines, the transformation matrix corresponds to applying the DCT, which can be performed in $\Theta(n\log n)$ time and $\Theta(n)$ memory using the Fast Fourier Transform algorithm [15], where n is the number of cells in the rectangle, which is proportional to its volume. For general shapes, we would get arbitrary basis functions, and these requirements would increase to $\Theta(n^2)$ in compute and memory, which quickly becomes prohibitive for large scenes, with n ranging in millions. Re-interpreting equation (1) in a discrete-space setting, substituting P from the expression above and re-arranging, we get,

$$\begin{aligned} \frac{\partial^2 M_i}{\partial t^2} + c^2 k_i^2 M_i &= DCT(F(t)), \\ k_i^2 &= \pi^2\left(\frac{i_x^2}{l_x^2}+\frac{i_y^2}{l_y^2}+\frac{i_z^2}{l_z^2}\right) \end{aligned}. \tag{6}$$

In the absence of any forcing term, the above equation describes a set of independent simple harmonic oscillators, with each one vibrating with its own characteristic frequency, $\omega_i = ck_i$. The above analysis may be equivalently regarded as Modal Analysis applied to a rectangular domain. However, our overall approach is different from Modal Analysis because the latter is typically applied to a domain as a whole, yielding arbitrary basis functions which do not yield to efficient transforms, and extracting all the modes is typically intractable for domains with millions of cells.

We model arbitrary forcing functions, for example, due to a volume sound sources as follows. Assuming that the forcing function, $F(t)$ is constant over a time-step Δt, it may be transformed to mode-space as –

$$\widetilde{F}(t) \equiv DCT(F(t)) \tag{7}$$

and one may derive the following update rule –

$$M_i^{n+1} = 2M_i^n\cos(\omega_i\Delta t) - M_i^{n-1} + \frac{2\widetilde{F^n}}{\omega_i^2}(1-\cos(\omega_i\Delta t)). \tag{8}$$

This update rule is obtained by using the closed form solution of a simple harmonic oscillator over a time-step. Since it is a second-order equation, we need to specify one more initial condition, which we choose to be that the function computed over the time-step evaluates correctly to the value at the previous time-step, M^{n-1}. This leads to a time-stepping scheme which has no numerical errors for propagation in the interior of the rectangle, since we are directly using the closed-form solution for a simple harmonic oscillator. The only error introduced is in assuming that the forcing term is constant over a time-step. This is not a problem for input source sounds, as the time-step is necessarily below the sampling rate of the input signal. However, the communication of sound between two rectangular domains is ensured through forcing terms on their interface and this approximation introduces numerical errors at the interface. We discuss these issues in detail in the next section.

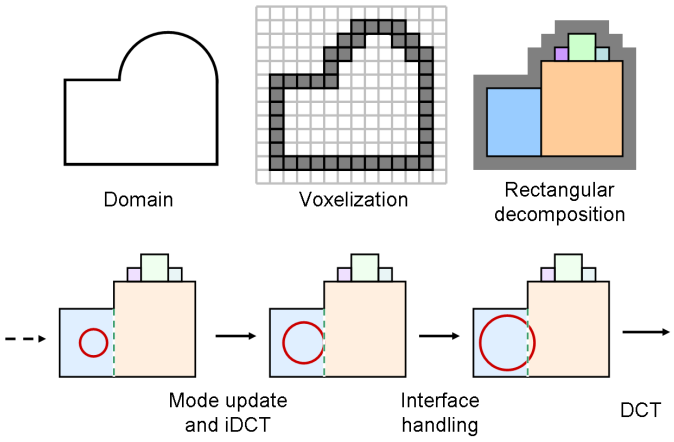


Fig. 3. Overview of our approach. The scene is first voxelized at a prescribed cell size depending on the highest simulated frequency. A rectangular decomposition is then performed and impulse response calculations then carried out on the resulting partitions. Each step is dominated by DCT and inverse DCT calculations withing partitions, followed by interface handling to communicate sound between partitions.

4 Technique

In the previous section, we discussed the errors and efficiency issues of the FDTD method and discussed a method to carry out numerical solution of the Wave Equation accurately and efficiently on rectangular domains. In this section, we discuss our technique which exploits these observations to perform acoustic simulation on arbitrary domains by decomposing them into rectangular partitions. We end with a discussion of the numerical errors in our approach.

4.1 Rectangular Decomposition

Most scenes of interest for the purpose of acoustic simulation necessarily have large empty spaces in their interior. Consider a large scene like, for example, a $30m$ high cathedral in which an impulse is triggered near the floor. With FDTD, this impulse would travel upwards and would accumulate numerical dispersion error at each cell it crosses. Given that the spatial step size is comparable to the wavelength of the impulse, which is typically a few centimeters, the impulse accumulates a lot of error, crossing hundreds to thousands of cells. In the previous section, we discussed that wave propagation on a rectangular domain can be performed very efficiently while introducing no numerical errors. If we fit a rectangle in the scene extending from the bottom to the top, the impulse would have no propagation error. This is the chief motivation for Rectangular Decomposition – Since there are large empty spaces in typical scenes, a decomposition of the space into rectangular partitions would yield many partitions with large volume and exact propagation could be performed in the interior of each.

We perform the rectangular decomposition by first voxelizing the scene. The cell size is chosen based on the maximum frequency to be simulated, as discussed previously. Next, the rectangular decomposition is performed using a greedy heuristic, which tries to find the largest rectangle it can grow from a random seed cell until all free cells are exhausted. We note here that the correctness of our technique does not depend on the optimality of the rectangular decomposition. A slightly sub-optimal partitioning with larger interface area affects the performance only slightly, as the interface area is roughly proportional to the surface area of the domain, while the runtime performance is dominated by the cost of DCT, which is performed on input proportional to the volume of the domain.

4.2 Interface Handling

Once the domain of interest has been decomposed into many rectangles, propagation simulation can be carried out inside each rectangle as described in Section 3.4. However, since every rectangle is assumed

to have perfectly reflecting walls, sound will not propagate across rectangles. We next discuss how this communication is performed using a Finite Difference approximation. Without loss of generality, lets assume that two rectangular partitions share an interface with normal along the X-axis. Recall the discussion of FDTD in Section 3.2. Assume for the moment that (2,6) FDTD is running in each rectangular partition, using the stencil given in equation (2) to evaluate $\frac{d^2p_i}{dx^2}$. Further, assume that cell i and $i+1$ are in different partitions and thus lie on their interface. As mentioned previously, Neumann boundary condition implies even symmetry of the pressure field about the interface and each partition is processed with this assumption. Thus, the Finite Difference stencil may also be thought of as a sum of two parts – The first part assumes that the pressure field has even symmetry about the interface, namely, $p_i = p_{i+1}, p_{i-1} = p_{i+2}$ *and* $p_{i-2} = p_{i+3}$, and this enforces Neumann boundary conditions. The residual part of the stencil accounts for deviations from this symmetry, cause by the pressure in the neighboring partition. Symbolically, representing the Finite Difference stencil in equation (2) as S–

$$S_i = S_i^0 + S_i', \textit{ where}$$
$$S_i^0 = \frac{1}{180h^2}\left(2p_{i-3} - 25p_{i-2} + 243p_{i-1} - 220p_i\right)$$
$$S_i' = \frac{1}{180h^2}\left(-2p_{i-2} + 27p_{i-1} - 270p_i + 270p_{i+1} - 27p_{i+2} + 2p_{i+3}\right).$$

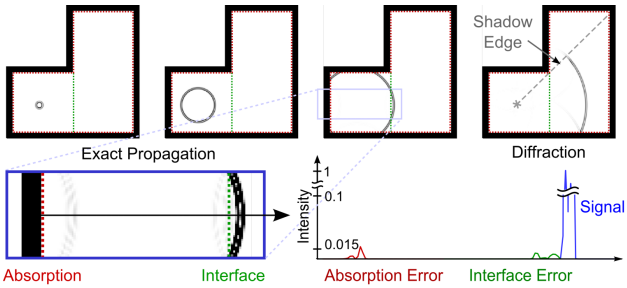
Since S_i' is a residual term not accounted for while evaluating the LHS of equation (3), it is transferred to the RHS and suitably accounted for in the forcing term, thus yielding,

$$F_i = c^2 S_i'. \tag{9}$$

Similar relations for the forcing term may be derived for all cells near the partition boundary which index cells in neighboring partitions. If we were actually using (2,6) FDTD in each partition, this forcing term would be exact, with the same numerical errors due to spatial and temporal approximations appearing in the interior as well as the interface. However, because we are using an exact solution in the interior, the interface handling described above introduces numerical errors equivalent to a (2,6) FDTD on the interface. We will discuss these errors in more detail shortly. We would like to note here that a sixth-order scheme was chosen as it gives sufficiently low interface errors, while being reasonably efficient. Lower (second/fourth) order schemes would be more efficient and much easier to implement, but as we have experimented, they result in much higher errors, which results in undesirable, audible high frequency noise. One may use an even higher order scheme if more accuracy is required for a particular application, at the expense of computation and implementaion effort. An interesting point to note at this point is that the interface handling doesn't need to know how the field inside each partition is being updated. Therefore, it is easy to mix different techniques for wave propagation in different parts of the domain, if so required.

4.3 Absorbing Boundary Condition

Our discussion till this point has assumed that all scene boundaries are perfectly reflecting. For modeling real scenes, this is an unrealistic assumption. Moreover, since the computation is carried out on a volumetric grid, it is necessary to truncate the domain and model emission into free space. It is necessary to have absorbing boundaries for this purpose. For this work, we have implemented the Perfectly Matched Layer (PML) absorber [34], which is commonly employed in most numerical wave propagation simulations due to its high absorption efficiency. PML works by applying an absorbing layer which uses coordinate stretching to model wave propagation in an unphysical medium with very high absorption, while ensuring that the impedance of the medium matches that of air at the interface to avoid reflection errors. The interfacing between the PML medium and a partition in our method is simple to implement – Since PML explicitly maintains a pressure field in the absorbing medium, the PML medium can also be treated as a partition and the same technique described above can be applied for the coupling between PML and other partitions. Variable reflectivity can be easily obtained by multiplying the forcing term



Frequency	Interface error	Absorption error
250 Hz	-54 dB	-21 dB
500 Hz	-42 dB	-26 dB
750 Hz	-36 dB	-26 dB
1000 Hz	-28 dB	-32 dB

Fig. 4. Measurements of numerical error due to interface handling and PML absorbing boundary conditions. The interface handling errors stays near -40 dB for most of the frequency spectrum, which is not perceptible. The absorption error stays around -25 dB which introduces very small errors in the reflectivity of different materials.

calculated for interface handling by a number between 0 and 1, 0 corresponding to full reflectivity and 1 corresponding to full absorption.

4.4 Putting everything together

In this subsection, we give a step-by-step description of all the steps involved in our technique. Figure 3 shows a schematic diagram of the different steps in our approach, which are as follows –

1. **Pre-processing**
   (a) Voxelize the scene. The cell-size is fixed by the minimum simulated wavelength and the required number of spatial samples per wavelength (typically 2-4)
   (b) Perform a rectangular decomposition on the resulting voxelization, as described in Section 4.1.
   (c) Perform any necessary pre-computation for the DCTs to be performed at runtime. Compute all interfaces and the partitions that share them.
2. **Simulation Loop**
   (a) Update modes within each partition using equation (8)
   (b) Transform all modes to pressure values by applying an iDCT as given in equation (5)
   (c) Compute and accumulate forcing terms for each cell. For cells on the interface, use equation (9), and for cells with point sources, use the sample value.
   (d) Transform forcing terms back to modal space using a DCT as given in equation (7).

4.5 Numerical Errors

Numerical errors in our method are introduced mainly through two sources – boundary approximation and interface errors. Since we employ a rectangular decomposition to approximate the simulation domain, there are stair-casing errors near the boundary (see Figure 7). These stair-casing errors are identical to those in FDTD because we do a rectangular decomposition of a uniform grid – there is no additional geometry-approximation error due to using rectangular partitions. In most room acoustic software, it is common practice to approximate the geometry to varying degrees [42]. The reason for doing this is that we are not as sensitive to acoustic detail as much as we are to visual detail. Geometric features comparable or smaller than the wavelength of sound (34 cm at 1kHz) lead to very small variations in the overall

acoustics of the scene due to the presence of diffraction. In contrast, in light simulation, all geometric details are visible because of the ultra-small wavelength of light and thus stair-casing is a much more important problem.

The net effect of stair-casing error for numerical simulators is that for frequencies with wavelengths comparable to the cell size (1kHz), the walls act as diffuse instead of specular reflectors. For frequencies with large wavelengths (500 Hz and below), the roughness of the surface is effectively 'invisible' to the wave, and the boundary errors are small with near-specular reflections. Therefore, the perceptual impact of boundary approximation is lesser in acoustic simulation.

However, if very high boundary accuracy is critical for a certain scene, this can be achieved by coupling our approach with a high-resolution grid near the boundary, running FDTD at a smaller time-step. As we had mentioned earlier, as long as the pressure values in neighboring cells are available, it is easy to couple the simulation in the rectangular partitions with another simulator running in some other part of the domain. Of course, this would create extra computational overhead, so its an efficiency-accuracy tradeoff.

As we discussed theoretically in Section 3.4 and also demonstrate with experiments in the next section, our technique is able to nearly eliminate numerical dispersion errors. However, because the inter-partition interface handling is based on a less accurate (2,6) FDTD scheme, the coupling is not perfect, which leads to erroneous reflections at the interface. Figure 4 shows the interface error for a simple scene. The Nyquist frequency on the mesh is 2000Hz. The table at the bottom of the figure shows the interface reflection errors for different frequencies, in terms of sound intensity. Although the interface errors increase with increasing frequency, it stays $\sim -40dB$ for most of the spectrum. Roughly, that is the difference in sound intensity between normal conversation and a large orchestra.

Since most scenes of practical interest have large empty spaces in their interior, the number of partition interfaces encountered by a wave traveling the diameter of the scene is quite low. For example, refer to Figure 7 – a wave traveling the 20 m distance from the source location to the dome at the top encounters only about 10 interfaces. Also, it is important to note that this is a worst-case scenario for our approach, since many rectangles are needed to fit the curved dome at the top. This is the chief advantage of our approach – numerical dispersion is removed for traveling this distance and it is traded off for very small reflection errors which are imperceptible. Please hear the accompanying video for examples of audio rendered on complex scenes with numerous interfaces.

Figure 4 also shows the absorption errors for the PML Absorbing Boundary Condition. The absorption errors range from -20 to -30dB, which works well for most scenes, since this only causes a slight deviation from the actual reflectivity of the material being modeled. However, if higher accuracy absorption is required, one might increase the PML thickness. We have used a 5-cell thick PML in all our simulations.

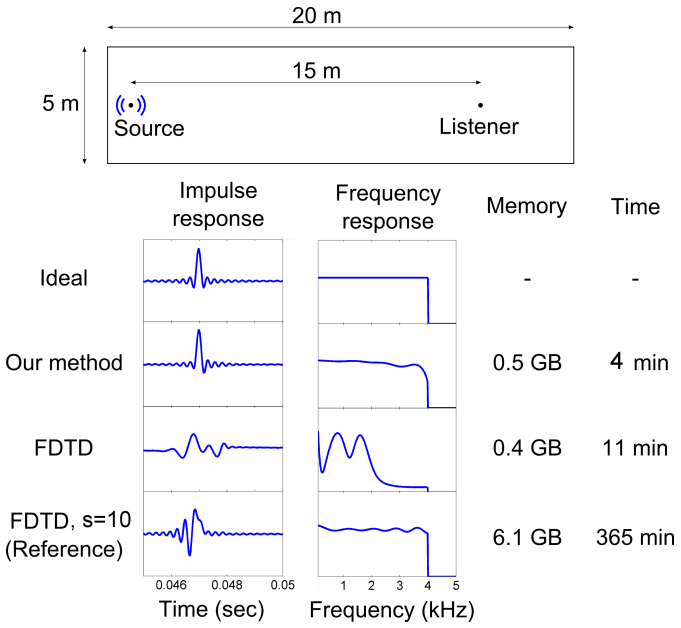


Fig. 5. Numerical results on the corridor scene, comparing numerical dispersion errors in FDTD and in our method. The reference FDTD solution has a mesh with $s = 10$ samples per wavelength. Note that only the magnitudes of the Fourier coefficients are plotted. Our method suffers from very little numerical dispersion, reproducing the ideal impulse response very closely, while FDTD suffers from large amounts for numerical dispersion. We take an order of magnitude less memory and nearly two orders of magnitude less computation time to produce results with accuracy comparable to the reference solution.

5 Results

5.1 Sound Rendering

The input to all audio simulations we perform is a Gaussian-derivative impulse of unit amplitude. Given the maximum frequency to be simulated, ν_{max}, we fix the width of the impulse so that its maxima in frequency domain is at $\frac{\nu_{max}}{2}$, giving a broadband impulse in the frequency range of interest. This impulse is triggered at the source location and simulation performed until the pressure field has dropped off to about -40 dB, which is roughly the numerical error of the simulation. The resulting signal is recorded at the listener position(s). Next, deconvolution is performed using a simple Fourier coefficient division to obtain the Impulse Response (IR), which is used for sound rendering at a given location.

Auralizing the sound at a moving listener location is performed as follows. First, note that running one simulation from a source location yields the pressure variation at all cell centers because we are solving for the complete field on a volume grid. For auralizing sound, we first compute the IRs at all cells lying close to the listener path. Next, the sound at the current position and time is estimated by linearly interpolating the field values at neighboring cell centers. To obtain the field value at a given cell center, a convolution of the IR at the corresponding location and the input sound is performed. We would like to emphasize here that there are more efficient ways of implementing the auralization but that is not the focus of this paper.

Most of the simulations we have performed are band-limited to 1-2kHz due to computation and memory constraints. However, this is not a big limitation. Although audible sounds go up to 22kHz, it is important to realize that only frequencies up to about 5kHz are perceptually critical [24] for acoustics simulation. Moreover, the frequency perception of humans is logarithmic, which reflects in the frequency doubling between musical octaves. This means that most of the perceptually important frequencies are contained till about 2kHz. For example, the frequency of a typical 88-key piano goes from about 30Hz to 4kHz, covering 7 octaves, out of which 6 octaves are below 2kHz. However, even though we don't have accurate perception of higher frequencies, their complete absence leads to perceptual artifacts and therefore, there must be some way of accounting for higher frequencies, even if approximately. One way of doing that would be to combine our technique with a Geometrical Acoustic simulator for the higher frequency range. In this paper, we have used a much simpler technique that gives good results in practice.

To auralize sounds in the full audible range up to 22kHz, we first upsample the IR obtained from the simulation to 44kHz and run a simple peak detector on the IR which works by searching for local maxima/minima. The resulting IR contains peaks with varying amplitudes and delays, corresponding to incoming impulses. This is exactly the kind of IR that geometrical approaches compute by tracing paths for sound and computing the attenuation and delay along different paths. Each path yields a contribution to the IR. The difference here is that numerical simulation does not explicitly trace these paths. Instead, we extract this information from the computed impulse response through peak detection. We use an IR thus computed for higher frequencies.

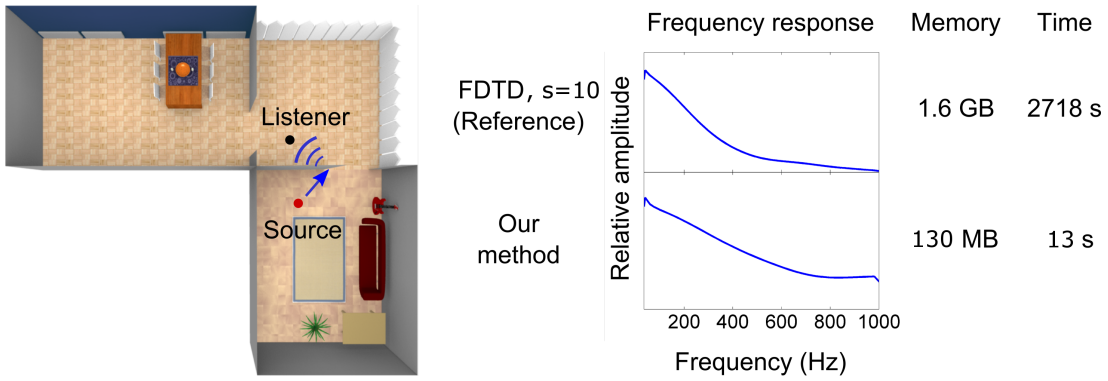


Fig. 6. The House scene demonstrates diffraction of sound around obstacles. All the scene geometry shown was included in the simulation. Our method is able to reproduce the higher diffraction intensity of sound at lower frequencies, while reducing the memory requirements by about an order of magnitude and the computational requirements by more than two orders of magnitude. The reference solution is computed on a mesh with $s = 10$ samples per wavelength.

The approximation introduced in this operation is that the diffraction at higher frequencies is approximated since the peak detector doesn't differentiate between reflection and diffraction peaks. Intuitively, this means that high frequencies may also diffract like low frequencies, which is the approximation introduced by this technique. This IR filter is then high-passed at the simulation cutoff frequency to yield a filter to be used exclusively for higher frequencies. As a final step, the exact low-frequency IR and approximate high-frequency IR are combined in frequency domain to yield the required IR to be applied on input signals. We must emphasize here that this technique is applied to obtain an approximate response exclusively in the high-frequency range and it is ensured that numerical accuracy for lower frequencies till 1-2kHz is maintained.

The reference solution for comparing our solution is the (2,6) FDTD method described in Section 3.2 running on a fine mesh that ensures 10 samples per wavelength. Since the main bottleneck of our approach is DCT, which can be performed through an FFT, we used the GPU-based FFT implementation described in [17], to exploit the compute power available on today's high-end graphics cards. Combining optimized transforms with algorithmic improvements described in the paper is the reason we gain considerable speedups over FDTD. All the simulations were performed on a 2.8GHz Intel Xeon CPU, with 8GB of RAM. The FFTs were performed on an NVIDIA GeForce GTX 280 graphics card.

In the following sub-sections, to clearly demarcate the algorithmic gain of our appoach over FDTD and the speedups obtained due to using the GPU implementation of FFT, we provide three timings for each case: the running time for computing the reference solution with FDTD, the time if we use a serial version of FFTW [15] and the time with the GPU implementation of FFT. In general, we obtain a ten-fold performance gain due to algorithmic improvements and another ten-fold due to using GPU FFT. The ten-fold gain in memory usage is of course, purely due to algorithmic improvements.

5.2 Numerical Dispersion: Anechoic Corridor

We first demonstrate the lack of numerical dispersion in our scheme. Refer to Figure 5. The scene is a $20m \times 5m \times 5m$ corridor in which the source and listener are located $15m$ apart, as shown in the figure. To measure just the accumulation of numerical dispersion in the direct sound and isolate any errors due to interface or boundary handling, we modeled the scene as a single, fully reflective rectangle. The simulation was band-limited to 4kHz, and the IR was calculated at the listener and only the direct sound part of the impulse response was retained. As Figure 5 shows, our method's impulse response is almost exactly the same as the ideal response. FDTD running on the same mesh undergoes large dispersion errors, while FDTD running on a refined mesh with s=10 samples per wavelength, (the reference) gives reasonably good results. Note that since there is only direct transmission from the source to the listener, the magnitude of the ideal frequency response is constant over all frequencies. This is faithfully observed for our method and the reference, but FDTD undergoes large errors, especially for high frequencies. Referring to the video, this is the reason that with FDTD, the sound is 'muffled' and dull, while with the reference solution and our technique, the consonants are clear and 'bright'. Therefore, as clearly demonstrated, our method achieves competitive accuracy with the reference while consuming 12 times less memory. The reference solution takes 365 minutes to compute, our technique with FFTW takes 31 minutes and our technique with GPU FFT takes about 4 minutes.

5.3 House Scene

It is a physically-observed phenomenon that lower frequencies tend to diffract more around an obstacle than higher frequencies. To illustrate that the associated gradual variation in intensity is actually observed with our method, we modeled a House scene, shown in Figure 6. Please listen to the accompanying video to listen to the corresponding sound clip. Initially, the listener is at the upper-right corner of the figure shown, and the sound source at the lower-left corner of the scene. The source is placed such that initially, there is no reflected path from the source to the listener. As the listener walks and reaches the door of the living room, the sound intensity grows gradually, instead of undergoing an unrealistic discontinuity as with geometric techniques which don't account explicitly for diffraction. This shows qualitatively that diffraction is captured properly by our simulator.

The dimensions of the House are $17m \times 15m \times 5m$ and the simulation was carried out till 2kHz. The wall reflectivity was set to 50%. The acoustic response was computed for .4 seconds. The total simulation time on this scene for the reference is about 3.5 days, 4.5 hours with our technique using FFTW and about 24 minutes with our technique using GPU FFT. The simulation takes about 700 MB of memory with our technique. This corresponds to speedups of about 18x due to algorithmic improvements and an additional 11x due to using GPU FFT.

To validate the diffraction accuracy of our simulator, we placed the source and listener as shown in Figure 6, such that the dominant path from the source to the listener is around the diffracting edge of the door. The middle of the figure shows a comparison of the frequency response (FFT of the Impulse Response) at the listener location, between the reference (FDTD on a fine mesh with s=10 samples per wavelength) and our solution. Note that both responses have a similar downward trend. This corroborates with the physically observed fact that lower frequencies diffract more than higher frequencies. Also, the two responses agree quite well. However, the slight discrepancy at higher frequencies is explained by the fact that there are two partition interfaces right at the diffraction edge and the corresponding interface errors result in the observed difference. Referring to Figure 6, observe

that our method takes 12x less memory and 200x less computation than the reference to produce reasonably accurate results.

5.4 Cathedral Scene

As our largest benchmark, we ran our sound simulator on a Cathedral scene (shown in Figure 1) of size $35m \times 26m \times 15m$. The simulation was carried out till 1kHz. The impulse response was computed for 2 seconds with absorptivity set to 10% and 40%, consuming less than 1GB of memory with our technique. We could not run the reference solution for this benchmark because it would take approximately 25GB of memory, which is not available on a desktop systems today, with a projected 2 weeks of computation for this same scene. The running times for this case are: 2 weeks for the reference (projected), 14 hours with our technique using FFTW and 58 minutes with our technique using GPU FFT. This scenario highlights the memory and computational efficiency of our approach, as well as a challenging case that the current approaches cannot handle on desktop workstations. Figure 7 shows the rectangular decomposition of this scene. Observe that our heuristic is able to fit very large rectangles in the interior of the domain. The main advantage of our approach in terms of accuracy is that propagation over large distances within these rectangles is error-free, while an FDTD implementation would accumulate dispersion errors over all cells a signal has to cross. The bottom of the figure shows the impulse response of the two simulations with low and high absorptivity in dB. Note that in both cases, the sound field decays exponentially with time, which is as expected physically. Also, with 40% absorption, the response decays much faster as compared to 10% absorption, decaying to -60 dB in 0.5 seconds. Therefore in the corresponding video, with low absorption, the sound is less coherent and individual notes are hard to discern, because strong reverberations from the walls interfere with the direct sound. This is similar to what is observed in cathedrals in reality.

Also note that we are able to capture high order reflections, corresponding to about 30 reflections in this scene. This *late reverberation* phase captures the echoic trail-off of sound in an environment. Geometric techniques typically have considerable degradation in performance with the order of reflections and are therefore usually limited to a few reflections. We are able to capture such high order reflections because of two reasons: Firstly, we are using a numerical technique which works directly with the volumetric sound field and is thus insensitive to the number of reflections. Secondly, as discussed in Section 5.2, our technique has very low numerical dispersion and thus preserves the signal well over long distances. For 30 reflections in the Cathedral, the signal must travel about 600 meters without much dispersion. As discussed earlier, with FDTD running on the same mesh, the signal would be destroyed in about 20 meters.

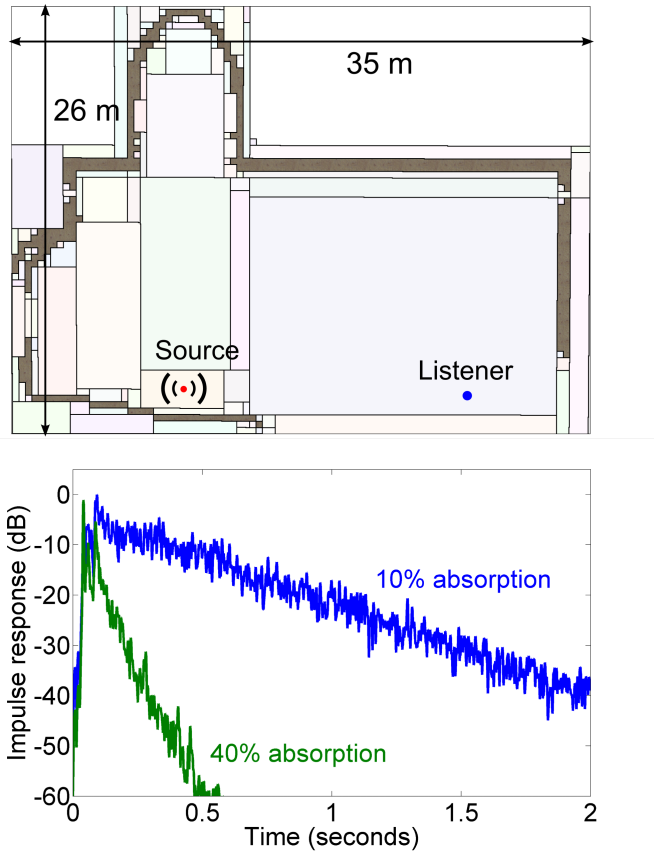


Fig. 7. The voxelization and rectangular decomposition of the Cathedral scene. Varying the absorptivity of the Cathedral walls directly affects the reverberation time. Note that we are able to capture all reflections in the scene, including later reverberation. The impulse responses shown above correspond to high order reflections, in the range of 30 reflections, which would be prohibitively expensive to compute accurately for geometric approaches.

6 Conclusion and Future Work

We have presented a computation- and memory-efficient technique for performing accurate numerical acoustic simulations on complex domains with millions of cells, for sounds in the kHz range. Our method exploits the analytical solution to the Wave Equation in rectangular domains and is at least an order of magnitude more efficient, both in terms of memory and computation, compared to a reference (2,6) FDTD scheme. Consequently, we are able to perform physically accurate sound simulation, which yields perceptually convincing results containing physical effects such as diffraction. With our technique, we have been able to perform numerical sound simulations on large, complex scenes, which, to the best of our knowledge, was not previously possible on a desktop computer.

One of the areas where our implementation may be improved is to add a fine-grid simulation near the boundary to reduce boundary reflection errors. Further, we are actively looking into the integration of stereo sound in our framework, which requires the ability to model dynamic objects in the scene. Also, we would like to model both moving sound sources and listener in the future. Another direction this work may be extended is to combine it with a geometric technique for performing the high-frequency part of the simulation, while our technique simulates frequencies up to 1-2 kHz.

REFERENCES

[1] Domain decomposition method. http://www.ddm.org.

[2] Soundscapes in half-life 2, valve corporation. http://developer.valvesoftware.com/wiki/Soundscapes, 2008.

[3] J. B. Allen and D. A. Berkley. Image method for efficiently simulating small-room acoustics. *J. Acoust. Soc. Am*, 65(4):943–950, 1979.

[4] F. Antonacci, M. Foco, A. Sarti, and S. Tubaro. Real time modeling of acoustic propagation in complex environments. *Proceedings of 7th International Conference on Digital Audio Effects*, pages 274–279, 2004.

[5] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, and H. Hagen. Phonon tracing for auralization and visualization of sound. In *IEEE Visualization 2005*, 2005.

[6] N. Bonneel, G. Drettakis, N. Tsingos, I. V. Delmon, and D. James. Fast modal sounds with scalable frequency-domain synthesis, August 2008.

[7] D. Botteldooren. Acoustical finite-difference time-domain simulation in a quasi-cartesian grid. *The Journal of the Acoustical Society of America*, 95(5):2313–2319, 1994.

[8] D. Botteldooren. Finite-difference time-domain simulation of low-frequency room acoustic problems. *Acoustical Society of America Journal*, 98:3302–3308, December 1995.

[9] J. P. Boyd. *Chebyshev and Fourier Spectral Methods : Second Revised Edition*. Dover Publications, December 2001.

[10] P. T. Calamia and P. U. Svensson. Fast time-domain edge-diffraction calculations for interactive acoustic simulations. *EURASIP Journal on Advances in Signal Processing*, 2007, 2007.

[11] C. A. de Moura. Parallel numerical methods for differential equations - a survey.

[12] E. Deines, F. Michel, M. Bertram, H. Hagen, and G. Nielson. Visualizing the phonon map. In *Eurovis*, 2006.

[13] Y. Dobashi, T. Yamamoto, and T. Nishita. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. *ACM Trans. Graph.*, 22(3):732–740, July 2003.

[14] Durlach. Virtual reality scientific and technological challenges. Technical report, National Research Council, 1995.

[15] M. Frigo and S. G. Johnson. The design and implementation of fftw3. *Proc. IEEE*, 93(2):216–231, 2005.

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

[17] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli. High performance discrete fourier transforms on graphics processors. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, Piscataway, NJ, USA, 2008. IEEE Press.

[18] M. Hodgson and E. M. Nosal. Experimental evaluation of radiosity for room sound-field prediction. *The Journal of the Acoustical Society of America*, 120(2):808–819, 2006.

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

[20] M. Karjalainen and C. Erkut. Digital waveguides versus finite difference structures: equivalence and mixed modeling. *EURASIP J. Appl. Signal Process.*, 2004(1):978–989, January 2004.

[21] L. E. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders. *Fundamentals of Acoustics*. Wiley, December 1999.

[22] M. Kleiner, B.-I. Dalenbäck, and P. Svensson. Auralization - an overview. *JAES*, 41:861–875, 1993.

[23] U. Krockstadt. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound Vibration*, 1968.

[24] H. Kuttruff. *Room Acoustics*. Taylor & Francis, October 2000.

[25] Q. H. Liu. The pstd algorithm: A time-domain method combining the pseudospectral technique and perfectly matched layers. *The Journal of the Acoustical Society of America*, 101(5):3182, 1997.

[26] T. Lokki. *Physically-based Auralization*. PhD thesis, Helsinki University of Technology, 2002.

[27] M. Monks, B. M. Oh, and J. Dorsey. Audioptimization: Goal-based acoustic design. *IEEE Computer Graphics and Applications*, 20(3):76 – 91, 2000.

[28] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley. Acoustic modeling using the digital waveguide mesh. *Signal Processing Magazine, IEEE*, 24(2):55–66, 2007.

[29] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181, New York, NY, USA, 2002. ACM.

[30] R. Petrausch and S. Rabenstein. Simulation of room acoustics via block-based physical modeling with the functional transformation method. *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, pages 195–198, 16-19 Oct. 2005.

[31] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999.

[32] R. Rabenstein, S. Petrausch, A. Sarti, G. De Sanctis, C. Erkut, and M. Karjalainen. Block-based physical modeling for digital sound synthesis. *Signal Processing Magazine, IEEE*, 24(2):42–54, 2007.

[33] N. Raghuvanshi and M. C. Lin. Interactive sound synthesis for large scale environments. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108, New York, NY, USA, 2006. ACM Press.

[34] Y. S. Rickard, N. K. Georgieva, and W.-P. Huang. Application and optimization of pml abc for the 3-d wave equation in the time domain. *Antennas and Propagation, IEEE Transactions on*, 51(2):286–295, 2003.

[35] J. H. Rindel. The use of computer modeling in room acoustics.

[36] H. Sabine. Room acoustics. *Audio, Transactions of the IRE Professional Group on*, 1(4):4–12, 1953.

[37] S. Sakamoto, T. Seimiya, and H. Tachibana. Visualization of sound reflection and diffraction using finite difference time domain method. *Acoustical Science and Technology*, 23(1):34–39, 2002.

[38] S. Sakamoto, A. Ushiyama, and H. Nagatomo. Numerical analysis of sound propagation in rooms using the finite difference time domain method. *The Journal of the Acoustical Society of America*, 120(5):3008, 2006.

[39] S. Sakamoto, T. Yokota, and H. Tachibana. Numerical sound field analysis in halls using the finite difference time domain method. In *RADS 2004*, Awaji, Japan, 2004.

[40] L. Savioja. *Modeling Techniques for Virtual Acoustics*. Doctoral thesis, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Report TML-A3, 1999.

[41] K. L. Shlager and J. B. Schneider. A selective survey of the finite-difference time-domain literature. *Antennas and Propagation Magazine, IEEE*, 37(4):39–57, 1995.

[42] S. Siltanen. Geometry reduction in room acoustics modeling. Master's thesis, Helsinki University of Technology, 2005.

[43] A. Taflove and S. C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method, Third Edition*. Artech House Publishers, June 2005.

[44] T. Takala and J. Hahn. Sound rendering. *SIGGRAPH Comput. Graph.*, 26(2):211–220, July 1992.

[45] A. Toselli and O. Widlund. *Domain Decomposition Methods*. Springer, 1 edition, November 2004.

[46] N. Tsingos. *Simulating High Quality Dynamic Virtual Sound Fields For Interactive Graphics Applications*. PhD thesis, Universite Joseph Fourier Grenoble I, December 1998.

[47] N. Tsingos, C. Dachsbacher, S. Lefebvre, and M. Dellepiane. Instant sound scattering. In *Rendering Techniques (Proceedings of the Eurographics Symposium on Rendering)*, 2007.

[48] N. Tsingos, T. Funkhouser, A. Ngan, , and I. Carlbom. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *Computer Graphics (SIGGRAPH 2001)*, August 2001.

[49] K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544, New York, NY, USA, 2001. ACM Press.

[50] S. Van Duyne and J. O. Smith. The 2-d digital waveguide mesh. In *Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on*, pages 177–180, 1993.

[51] K. Yee. Numerical solution of inital boundary value problems involving maxwell's equations in isotropic media. *Antennas and Propagation, IEEE Transactions on [legacy, pre - 1988]*, 14(3):302–307, 1966.