Proc. of the EAA Symposium on Auralization, Espoo, Finland, 15-17 June 2009

EFFICIENT NUMERICAL ACOUSTIC SIMULATION ON GRAPHICS PROCESSORS USING ADAPTIVE RECTANGULAR DECOMPOSITION

Nikunj Raghuvanshi,[*] Dept. of Computer Science, UNC Chapel Hill, USA

Brandon Lloyd,[†] Microsoft Corporation

Naga K. Govindaraju,[‡] Microsoft Corporation

Ming C.Lin,[§] Dept. of Computer Science, UNC Chapel Hill, USA

ABSTRACT

Accurate acoustic simulation can enable realistic auralization that leads to enhanced immersion for visual applications, as well as facilitates accurate predictions for practical room acoustic scenarios. Numerical simulation provides realistic impulse responses that properly account for interference and diffraction effects by modeling the physics of wave propagation. However, it has posed a tough computational challenge owing to its large computation and memory requirements. We present a technique which relies on an adaptive rectangular decomposition of 3D scenes that yields two key advantages: Firstly, its key computational routine is DCT which can be efficiently parallelized on Graphics Processors. Secondly, the analytical solution of the Wave Equation in rectangular domains is known, which can be exploited to gain in accuracy and perform simulations on coarse simulation meshes, reducing both the computation and memory requirements further. Our technique is able to achieve a gain of at least a hundred-fold in computation and ten-fold in memory compared to a standard Finite Difference Time Domain (FDTD) implementation with comparable accuracy.

1. INTRODUCTION

Efficient and accurate acoustic simulation can be used to calculate realistic impulse responses (IRs) without the need of capturing them on a real scene. In many cases, like virtual acoustics or auditorium design, the scene might not even exist in reality and such accurate predictions can add to the realism or predictive auralizations, depending on the application. Numerical Acoustic simulation is a promising way to obtain IRs for arbitrary scenes while taking into account complex wave-based acoustic phenomena like interference and diffraction that are required for realistic auralization.

The problem of acoustic simulation is challenging because audible sounds have wavelengths that falls in the range of the dimensions of common objects, and consequently sound diffracts appreciably around most objects, especially at frequencies up to around 1 kHz. Capturing diffraction has been a tough challenge for Geometric acoustic techniques. In addition, the speed of sound is small enough that a complete transient simulation needs to be performed in time-domain for practical auralization requirements. Computational acoustics thus has its own unique challenges.

[*] nikunjr@gmail.com
[†] dalloyd@microsoft.com
[‡] nagag@microsoft.com
[§] lin@cs.unc.edu

Since numerical approaches attempt to integrate the underlying Linear Wave Equation directly, they are capable of performing a full transient solution which correctly accounts for all wave phenomena, including diffraction, elegantly in one framework. This capability is their most attractive feature. But this accuracy comes at a very high computational cost and large memory requirements for practical scenes – especially so at medium to high frequencies above a few hundred Hz. The reason is that typically numerical simulation running time scales as the fourth power and memory requirement as the third power of the maximum simulated frequency. Naively reducing the grid resolution leads to unacceptably large numerical errors.

Main Results: Our technique is at least ten-fold efficient in memory and hundred-fold efficient in computation compared to a standard FDTD implementation and achieves competitive accuracy at the same time. It relies on an *adaptive rectangular decomposition* of the free space of the scene. This approach has two key advantages:

1. The solution to the Wave equation in a Rectangular domain can be be performed efficiently through a Discrete Cosine Transform (DCT) that can be calculated efficiently through FFT. We have integrated our simulator with an implementation of FFT on Graphics Processing Units (GPU) [1], to gain considerable speedups over CPU-based libraries.
2. The analytical solution to the wave equation within a rectangular domain is known. This enables highly reduced numerical dispersion errors, even on grids approaching the Nyquist limit with only two to four spatial samples per wavelength (compared to about ten samples per wavelength required by other numerical techniques like FDTD). This leads to significant gains in both computation and memory requirements.

For practical auralizations, it is required to numerically model partially absorbing surfaces as well as fully absorbing surfaces corresponding to open space, such as open doors and windows. We have implemented the Perfectly Matched Layer (PML) Absorbing Boundary Condition to handle such cases. We demonstrate our algorithm and resulting auralization on practical scenarios with high complexity and validate the results against FDTD. Owing to the computational efficiency of our approach, to the best of our knowledge, our work is the first to show accurate time-domain numerical acoustic simulation on such large, complex 3D scenes for all frequencies up to 1kHz on a high-end 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

some mathematical background, which motivates our approach described in Section 4. We discuss results obtained with our technique in Section 5.

2. PREVIOUS WORK

The ever-increasing computational power of desktop computers has enabled more and more accurate acoustic simulation and auralization on realistic scenes. The fundamental equation governing wave propagation for most practical cases is the Linear Wave Equation. Depending on how wave propagation is approximated, techniques for simulating acoustics may be broadly classified into Geometrical Acoustics (GA) and Numerical Acoustics (NA). We briefly survey work in both these areas.

Geometric Acoustics: GA approaches are derived using asymptotic approximations of the Wave Equation in the infinite frequency limit. Their basic assumption is that sound propagates in linear paths, like light. The first GA approaches to be investigated were the Image Source Method and Ray Tracing [2, 3]. Most room acoustics software use a combination of these techniques to this day [4]. Another efficient geometric approach proposed in literature is Beam Tracing [5, 6] or alternatively, Frustum Tracing [7] that assume continuous bundles of sound rays, instead of infinitely-thin rays. There has been work on Phonon Tracing [8] that assumes linearly-propagating packets of energy, called Phonons. Also, researchers have proposed applying hierarchical radiosity to acoustical energy transfer [9, 10]. Since all GA approaches inherently lack diffraction, it has to be accounted for explicitly. Most of such approaches rely on the Geometric Theory of Diffraction [11, 12, 13]. Diffraction remains a challenge for GA approaches and is an active area of research.

Numerical Acoustics: Numerical approaches differ in how they discretize and approximate the Wave Equation. Based on this, they may be 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 [14]. DWM approaches [15], on the other hand, use discrete waveguide elements forming a 3D cartesian mesh. Each waveguide is assumed to carry waves along its length in a single dimension. [16, 17, 18].

The FDTD method, owing to its simplicity and versatility, has been an active area of research in room acoustics for more than a decade [19]. 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 for low frequencies. It is only recently that the possibility of applying FDTD to medium sized scenes in 3D has been explored [20, 21]. Even then, the compute and memory requirements for FDTD are beyond the capability of most desktop systems today [21], requiring days of computation on a small cluster for medium-sized 3D scenes for simulating frequencies up to 1 kHz.

The Functional Transform Method (FTM) [22] is closely related to our technique. Although the mathematical frameworks of the two techniques are very different there are some similarities, like using a rectangular decomposition although the mathematical motivation is different. Our technique has the advantage of being very simple to formulate and works directly on the second order Wave Equation without casting it as a first order system which leads to gains in efficiency. Also, our technique requires just one mathematical transform, the DCT.

3. MATHEMATICAL BACKGROUND

We briefly discuss the mathematical background for our technique, as well as some details of the FDTD technique we compare against.

3.1. The Wave Equation

The input to an acoustics simulator is a scene in 3D, along with the boundary conditions and the locations of the sound source. The result is the entire sound field on a volumetric grid over time, which can be used to compute the impulse response at all possible listener locations in the scene.

The propagation of sound in a domain is governed by the Linear Acoustic Wave Equation,

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

Sound is represented as a time-varying spatial pressure field $p(\mathbf{x}, t)$. The speed of sound is kept as $c = 340ms^{-1}$ and $F(\mathbf{x}, t)$ is the forcing term that models 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.

3.2. Reference FDTD Scheme

FDTD works on a uniform axis-aligned grid with spatial spacing h. To capture the propagation of a prescribed maximum frequency ν_{max}, the Nyquist theorem requires that $h \leq \frac{\lambda_{max}}{2} = \frac{c}{2\nu_{max}}$. The Laplacian operator is then replaced with a discrete approximation of sixth order accuracy with the following numerical differentiation formula:

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

where p_i is the i^{th} grid cell in the corresponding dimension. The above expression can be compactly expressed in terms of the Discrete Laplacian Matrix, K and vectors P and F containing the pressures and source terms respectively, for all grid cells. Discretize Eqn. (1) in space as above and then in time at some time-step Δt, which is restricted by the CFL (Courant-Friedrich-Levy) condition $\Delta t < \frac{h}{c\sqrt{3}}$. Using the second-order Leapfrog integrator in time, the complete update rule is as follows:

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

Controlling numerical errors in FDTD typically requires not 2 (Nyquist) but 8-10 samples per wavelength [23]. This makes the method about $5^4 = 625$ times slower than if it had been running on a Nyquist mesh. The numerical errors lead to *Numerical Dispersion* – Waves with different frequenceis travel with different speeds. Reducing cell size increases accuracy, but is prohibitive as reducing the cell size by r times results in an increase in memory requirement by a factor of r^3 and total compute time by r^4.

4. TECHNIQUE

In this section, we briefly describe our technique to perform acoustic simulation on arbitrary domains.

4.1. Wave Equation and DCT

Consider a rectangular space in 3D, with solid diagonal extending from $(0,0,0)$ to (l_x, l_y, l_z), with perfectly reflective walls. Any pressure field $p(x,y,z,t)$ in this space can be represented as

$$p(x,y,z,t) = \sum_{i=(i_x,i_y,i_z)} m_i(t)\,\Phi_i(x,y,z), \tag{3}$$

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

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

In the discrete interpretation, Eqn. (3) is simply an inverse Discrete Cosine Transform (iDCT) in 3D, with Φ_i being the Cosine basis vectors for the given lengths. Therefore, we may efficiently transform from mode coefficient vector (M) to pressure vector (P) as –

$$P(t) = iDCT(M(t)). \tag{4}$$

The DCT and iDCT operations can be performed in $\Theta(n \log n)$ time and $\Theta(n)$ memory using the Fast Fourier Transform algorithm [24]. Re-interpreting Eqn. (1) in a discrete-space setting, substituting P from the expression above and re-arranging, we get,

$$\begin{aligned} \tfrac{\partial^2 m_i}{\partial t^2} + c^2 k_i^2 m_i &= iDCT(F(t)), \\ k_i^2 &= \pi^2\left(\tfrac{i_x^2}{l_x^2} + \tfrac{i_y^2}{l_y^2} + \tfrac{i_z^2}{l_z^2}\right) \end{aligned}. \tag{5}$$

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, $\omega_i = ck_i$. Assuming that $F(t)$ is constant over a time-step Δt, it may be transformed to mode-space as –

$$\widetilde{F}(t) \equiv DCT(F(t)) \tag{6}$$

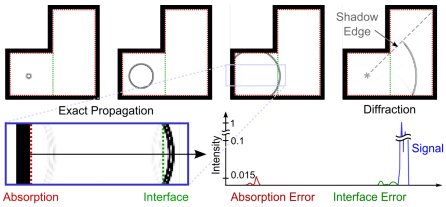
and one may derive the following update rule –

$$M_i^{n+1} = 2M_i^n \cos(\omega_i \Delta t) - M_i^{n-1} + \frac{2\widetilde{F^n}}{\omega_i^2}(1 - \cos(\omega_i \Delta t)). \tag{7}$$

This update rule is obtained by using the closed form solution of a simple harmonic oscillator over a time-step. Thus, the complete pressure field in a rectangular space can be updated in time by successive applications of Eqns. (6), (7) and (4), in that order, while incurring no numerical error except that inherent in DCT and errors in F.

4.2. Computing DCT on Graphics Processors (GPUs)

We now provide a brief summary of GPU-based DCT and its integration with our simulator. The details of this technique can be found in [1]. The GPU consists of a large number of scalar, in-order processors that can execute the same program in parallel using threads. Scalar processors are grouped together into multiprocessors. The GPU memory hierarchy is designed for high bandwidth to the global memory that is visible to all multiprocessors. The shared memory has low latency and is organized into several banks to provide higher bandwidth. At a high-level, computation on the GPU proceeds as follows: The user allocates memory on the GPU, copies data to the GPU, specifies a GPU program that executes on the multiprocessors and after execution, copies the data back to the host.



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

Figure 1: Numerical errors due to interface handling and PML absorber.

The forward Discrete Fourier Transforms (DFT) of a real sequence $x = x_0, \ldots, x_{N-1}$ is an N-point complex sequence that is conjugate symmetric, $X = X_0, \ldots, X_{N-1}$, where

$$X_k = \sum_{j=0}^{N-1} x_j e^{-2\pi i j k/N}, X_{N-1-i} = X_i{}^*. \tag{8}$$

The inverse DFT is defined as

$$x_k = \frac{1}{N}\sum_{j=0}^{N-1} X_j e^{2\pi i j k/N}. \tag{9}$$

FFT algorithms compute the DFT in O($N \log N$) operations. Using symmetries in the FFT of real data, the operation count can be reduced by half. A detailed overview of FFT algorithms can found in Van Loan [24].

The DCT can be computed using the FFT. The elements of a sequence are first permuted and then each element is scaled by an appropriate factor. Suppose that the length of the sequence is $N = 2p$. The sequence is permuted by placing the p even elements first in the new sequence before the odd elements, which appear in reverse order. Each element x_k is multiplied by $Re(e^{-2\pi i k/(4N)}) = \cos(-2\pi i k/(4N))$. The resulting sequence is transformed using the real FFT, yielding the DCT. The inverse DCT is computed using a similar process. See Van Loan [24] for details. For 3D DCTs as required in the discussion above, we perform the permutations simultaneously for all three dimensions.

We now describe the integration of our simulator with GPU-DCT. The simulator spawns two threads, one for the CPU and another for GPU processing. At each time-step, for each partition, the following operations are performed. In the GPU thread, the forcing coefficients are sent to the GPU and DCT invoked, as in Eqn. (6). Mode update performed as in Eqn. (7), and an inverse DCT performed as in Eqn. (4) to recover the pressure field in the partition. Meanwhile, the pressure field in each absorber partition is updated on the CPU thread in parallel. The details of absorber

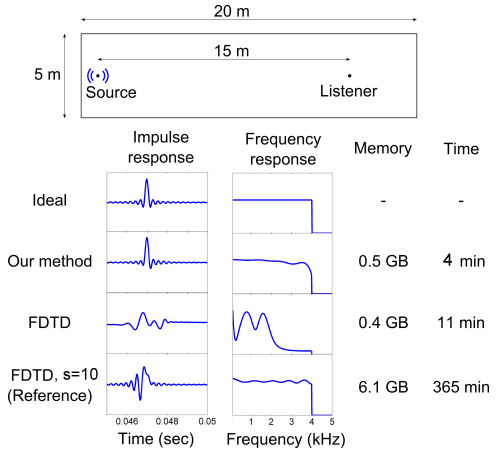


Figure 2: Numerical dispersion error in our method is comparable to that of the reference, while FDTD on the same mesh exhibits large error.

partitions will be discussed shortly. This serves to hide the cost of partially reflecting surfaces in the cost of propagation in the interior, using the GPU as an effective co-processor. After this, both threads are synchronized and interface handling performed on the CPU, which takes negligible time compared to the GPU processing.

4.3. Rectangular Decomposition and Interface Handling

Typical acoustic spaces are not rectangular, but can be always be partitioned into a set of rectangles touching each other. We perform this rectangular decomposition by first voxelizing the scene with a cell size that is chosen based on the maximum frequency to be simulated. Rectangles are fit using a randomized greedy heuristic which tries to "grow" the largest rectangle from the current seed point by successively increasing the length in each dimension by one. Once this can't be done further, the rectangle is stored, another random seed chosen and the process repeated until the free space is exhausted.

Acoustic simulation within each rectangular partition can be carried out as described above. However, interface operations need to be performed between partitions to propagate sound between them. The interface operators are based on a Finite Difference approximation. Assume two rectangular partitions share an interface with normal along the X-axis. Recall the discussion of FDTD in Section 3.2. Assuming, that cells i and $i+1$ are in different partitions and thus lie on a partition-partition interface. Using the the sixth-order Finite Difference stencil in Eqn. (2) the following interface operator may be derived –

$$S'_i = \frac{-2p_{i-2} + 27p_{i-1} - 270p_i + 270p_{i+1} - 27p_{i+2} + 2p_{i+3}}{180h^2} \tag{10}$$

This finite difference operator is accounted for in the forcing term, thus yielding,

$$F_i = c^2 S'_i. \tag{11}$$

Intuitively, the sound between two partitions is communicated using point sources on their shared interface. The numerical errors introduced due to interface handling will be discussed in detail shortly.

Absorbing Surfaces: To model partially reflecting surfaces, we have used the Perfectly Matched Layer (PML) absorber [25]. PML

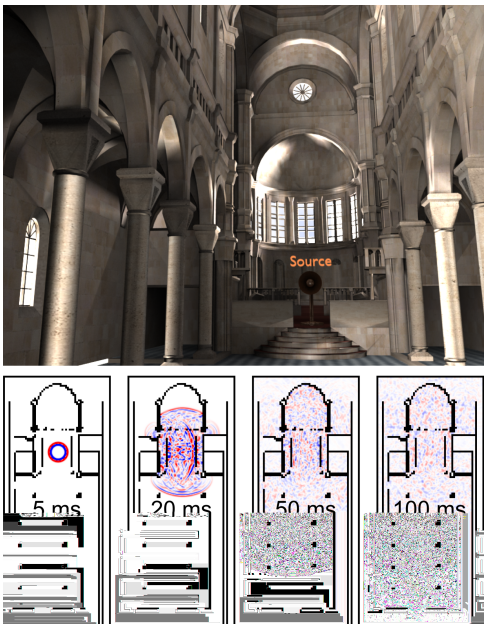


Figure 3: Numerical acoustic simulation and auralization on the Sibenik Cathedral. The dimensions of this cathedral are $35m \times 26m \times 15m$ with a volume of 13,650 m^3. The images at the bottom show an impulse propagating in the scene over time.

applies a thin absorber on the surface patch of interest and models a highly absorptive Wave Equation in the interior. The interfacing operator between the PML medium and a partition in our method is identical to that discussed above. Variable reflectivity is obtained by multiplying the forcing term calculated for interface handling by a number between 0 and 1, 0 corresponding to full reflectivity and 1 corresponding to full absorption.

Further details of this technique may be found in [26, 27].

5. RESULTS AND ANALYSIS

5.1. Numerical Errors

Figure 1 shows the interface error for a simple scene, which appear as fictitious reflections at the interface. Although the interface errors increase with increasing frequency, they stay near $-40dB$ for most of the spectrum. The figure also shows the absorption errors for the PML absorber, which ranges from -20 to -30dB, causing very slight deviations from the actual reflectivity of the material being modeled.

Since we employ a rectangular decomposition to approximate the simulation domain, their are stair-casing errors near the boundary (see Figure 4). The size of the stair-casing is necessarily below the smallest simulated wavelength causing only small errors. In the worst case, stair-casing makes the sound-field more diffuse than it should be. If very high boundary accuracy is critical, that can be achieved by coupling our approach with a fine-mesh simulation near the boundary.

5.2. Auralization

The input to all audio simulations we perform is a Gaussian derivative impulse with desired bandwidth (typically 1kHz). Auralizing the sound at a moving listener location is performed as follows. A simulation is run from the source location, yielding the pressure

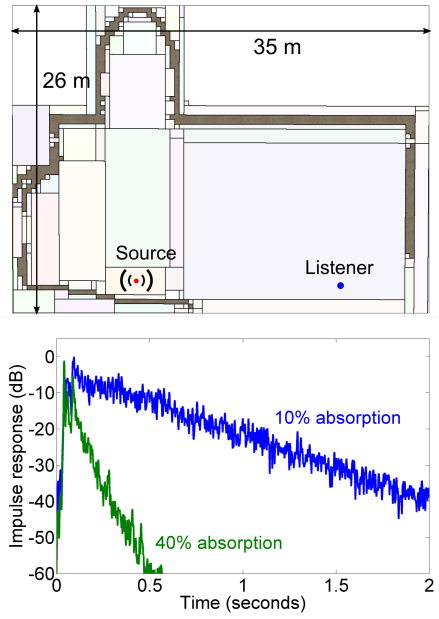


Figure 4: Visualization of rectangular decomposition of the Sibenik cathedral. Varying the absorptivity of the walls directly affects the reverberation time, as expected.

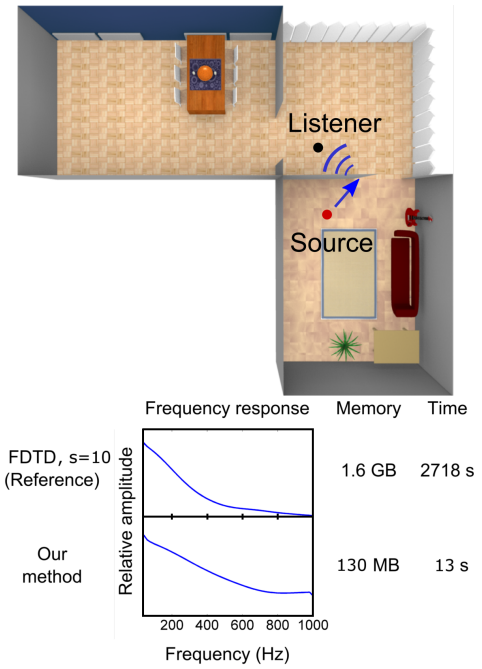


Figure 5: Diffraction of sound around edges. Our method reproduces the frequency domain low-passing effect of an edge.

signal at all cell centers. We then compute the IRs at all cells lying close to the listener path by performing a deconvolution by division in frequency domain. Next, the sound at the current position and time is estimated by linearly interpolating the sample values obtained by convolving the source signal with the IRs at the two nearest cell centers.

Most of the simulations we have performed are band-limited to 1-2kHz due to computation and memory constraints. However, this limitation can be partially removed for the purpose of auralization using a simple technique that we describe next. We first up-sample the IR obtained from the simulation to 44kHz by 0-padding in frequency domain and run a simple peak detector on the resulting IR. The peak detector works by searching for local maxima/minima and thus finds out significant reflection/diffraction peaks in the IR and their times. This IR is similar to that obtained with GA approaches and is used for frequencies above the maximum simulated frequency. 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. This means that high frequencies may also diffract like low frequencies.

The reference solution for comparing our solution is the FDTD method described in Section 3.2 running on a fine mesh that ensures 10 samples per wavelength (FDTD 2.5x). All the simulations were performed on a 2.8GHz Intel Xeon CPU, with 8GB of RAM. The GPU used was an NVIDIA GeForce GTX 280.

5.3. Numerical Dispersion: Anechoic Corridor

We first demonstrate the reduced numerical dispersion in our scheme. Refer to Figure 2. The scene is a $20m \times 5m \times 5m$ corridor with 6.5 million simulation cells in which the source and listener are located $15m$ apart, as shown in the figure. 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 2 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 2.5x refined mesh (the reference) gives reasonably good results. Our method achieves competitive accuracy with the reference while consuming 12 times less memory and 90 times less computation.

5.4. House Scene

To illustrate that diffraction and the associated gradual variation in intensity around an edge is actually observed, we modeled a House scene, shown in Figure 5. Please see the supplementary video for this auralization. Initially the listener is at the upper-right corner and the sound source at the lower right corner of the scene. The source is placed such that initially, diffraction is the dominant energy path from the source to the listener. As the listener walks and reaches the door of the living room, the sound intensity grows gradually. The dimensions of the House are $17m \times 15m \times 5m$, with 8.8 million simulation cells. The wall reflectivity was set to 50% and the simulation grid supported frequencies up to 4kHz. The acoustic response was computed for .4 seconds. The total simulation time on this scene for the reference is about 3.5 days and about 24 minutes with our technique. The simulation takes about 700 MB of memory with our technique and nearly 8 GB for the reference.

To validate the diffraction accuracy of our simulator, we placed the source and listener as shown in Figure 5, such that the dominant path from the source to the listener is only around the diffracting edge of the door. The middle of the figure shows a comparison of the frequency response for the first arriving peak at the listener location, between the reference and our solution. The two responses agree in their trend but there's a slight discrepancy at higher frequencies. A possible explanation is that there are two partition interfaces right at the diffraction edge and the corresponding interface errors result in the observed difference.

5.5. Cathedral Scene

As our largest benchmark, we ran our sound simulator on a the Sibenik cathedral scene (shown in Figure 3) of size $35m \times 26m \times 15m$, with 11.9 million simulation cells. The simulation was carried out till 1kHz on a mesh which supported up to 2kHz for a duration of 2 seconds. 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. With our technique it the simulation took about 58 minutes, consuming less than 1 GB of memory. An important point to note is that we are able to compute even the Late Reverberation phase easily because numerical techniques are insensitive to the order of reflection.

Figure 4 shows the rectangular decomposition of this scene. The bottom of the figure shows the impulse response of the two simulations with low and high absorptivity in dB against time. Note how in both cases, the Late Reverberation field decays exponentially with time, as expected physically.

6. CONCLUSION AND FUTURE WORK

We have presented a computation and memory-efficient technique for performing accurate numerical acoustic simulations on complex domains enabling auralization containing both diffraction as well as accurate Late Reverberation. We are actively looking into the integration of stereo sound in our framework, since numerical simulation does not provide directionality information explicitly. Also, we would like to add support for moving sound sources. 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.

7. REFERENCES

[1] Naga K. Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli, "High performance discrete fourier transforms on graphics processors," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Piscataway, NJ, USA, 2008, pp. 1–12, IEEE Press.

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

[3] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Am*, vol. 65, no. 4, pp. 943–950, 1979.

[4] J. H. Rindel, "The use of computer modeling in room acoustics," *Journal of Vibroengineering*, vol. 3, no. 4, pp. 219–224, 2000.

[5] Thomas Funkhouser, Nicolas Tsingos, Ingrid Carlbom, Gary Elko, Mohan Sondhi, James E. West, Gopal Pingali, Patrick Min, and Addy Ngan, "A beam tracing method for interactive architectural acoustics," *The Journal of the Acoustical Society of America*, vol. 115, no. 2, pp. 739–756, 2004.

[6] 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*, pp. 274–279, 2004.

[7] Anish Chandak, Christian Lauterbach, Micah Taylor, Zhimin Ren, and Dinesh Manocha, "Ad-frustum: Adaptive frustum tracing for interactive sound propagation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1707–1722, 2008.

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

[9] Nicolas Tsingos, *Simulating High Quality Dynamic Virtual Sound Fields For Interactive Graphics Applications*, Ph.D. thesis, Universite Joseph Fourier Grenoble I, December 1998.

[10] Murray Hodgson and Eva M. Nosal, "Experimental evaluation of radiosity for room sound-field prediction," *The Journal of the Acoustical Society of America*, vol. 120, no. 2, pp. 808–819, 2006.

[11] Nicolas Tsingos, Thomas Funkhouser, Addy Ngan, , and Ingrid Carlbom, "Modeling acoustics in virtual environments using the uniform theory of diffraction," in *Computer Graphics (SIGGRAPH 2001)*, August 2001.

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

[13] Micah Taylor, Anish Chandak, Zhimin Ren, Christian Lauterbach, and Dinesh Manocha, "Interactive edge diffraction for sound propagation in complex virtual environments," Tech. Rep., Department of Computer Science, UNC Chapel Hill, 2008.

[14] Mendel Kleiner, Bengt-Inge Dalenbäck, and Peter Svensson, "Auralization - an overview," *JAES*, vol. 41, pp. 861–875, 1993.

[15] 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*, 1993, pp. 177–180.

[16] Matti Karjalainen and Cumhur Erkut, "Digital waveguides versus finite difference structures: equivalence and mixed modeling," *EURASIP J. Appl. Signal Process.*, vol. 2004, no. 1, pp. 978–989, January 2004.

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

[18] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, "Acoustic modeling using the digital waveguide mesh," *Signal Processing Magazine, IEEE*, vol. 24, no. 2, pp. 55–66, 2007.

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

[20] Shinichi Sakamoto, Takuma Seimiya, and Hideki Tachibana, "Visualization of sound reflection and diffraction using finite difference time domain method," *Acoustical Science and Technology*, vol. 23, no. 1, pp. 34–39, 2002.

[21] 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.

[22] 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*, vol. 24, no. 2, pp. 42–54, 2007.

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

[24] Charles Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial Mathematics, 1992.

[25] Y. S. Rickard, N. K. Georgieva, and Wei-Ping Huang, "Application and optimization of pml abc for the 3-d wave equation in the time domain," *Antennas and Propagation, IEEE Transactions on*, vol. 51, no. 2, pp. 286–295, 2003.

[26] Nikunj Raghuvanshi, Nico Galoppo, and Ming C. Lin, "Accelerated wave-based acoustics simulation," in *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, New York, NY, USA, 2008, pp. 91–102, ACM.

[27] Nikunj Raghuvanshi, Rahul Narain, and Ming C. Lin, "Efficient and accurate sound propagation using adaptive rectangular decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 99, no. 1, 2009.