Parametric Wave Field Coding for Precomputed Sound Propagation

Nikunj Raghuvanshi John Snyder
Microsoft Research

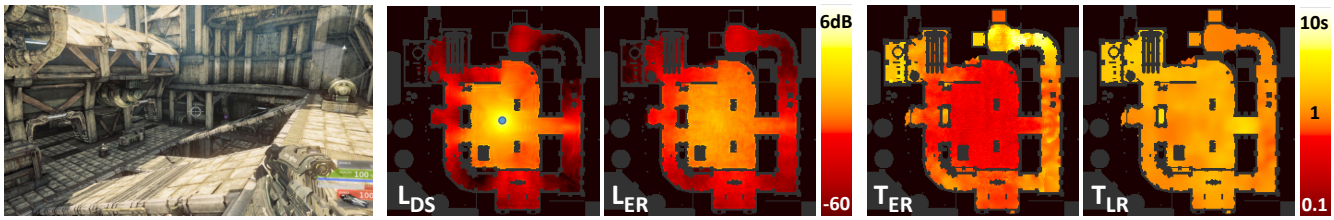


Figure 1: Our wave coding transforms 7D pressure fields (dependent on source/listener location and time) generated by numerical wave simulation to time-invariant 6D fields based on four perceptual parameters. Consistent with everyday experience, these parameters vary smoothly in space, aiding compression. Scene geometry ('Deck') is shown on the left, followed by a 2D slice of the parameter fields for a single source (blue dot). Direct sound loudness (L_{DS}) exhibits strong shadowing while early reflection loudness (L_{ER}) captures numerous scattered/diffracted paths, and consequently shadows less. Low L_{DS} combined with high L_{ER} conveys a distant and/or occluded source. Early decay time (T_{ER}) and late reverberation time (T_{LR}) together indicate scene size, reflectivity and openness. T_{LR} is spatially smoother than T_{ER}, being determined by many more weaker and higher-order paths in this complex space.

Abstract

The acoustic wave field in a complex scene is a chaotic 7D function of time and the positions of source and listener, making it difficult to compress and interpolate. This hampers precomputed approaches which tabulate impulse responses (IRs) to allow immersive, real-time sound propagation in static scenes. We code the field of time-varying IRs in terms of a few perceptual parameters derived from the IR's energy decay. The resulting parameter fields are spatially smooth and compressed using a lossless scheme similar to PNG. We show that this encoding removes two of the seven dimensions, making it possible to handle large scenes such as entire game maps within 100MB of memory. Run-time decoding is fast, taking 100μs per source. We introduce an efficient and scalable method for convolutionally rendering acoustic parameters that generates artifact-free audio even for fast motion and sudden changes in reverberance. We demonstrate convincing spatially-varying effects in complex scenes including occlusion/obstruction and reverberation, in our system integrated with Unreal Engine 3^{TM}.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Modeling;

Keywords: wave equation, diffraction, scattering, room acoustics, early decay time, reverberation time, occlusion, obstruction, exclusion, parametric reverb, impulse response, convolution, environmental effects, DSP

Links: DL PDF

ACM Reference Format
Raghuvanshi, N., Snyder, J. 2014. Parametric Wave Field Coding for Precomputed Sound Propagation. ACM Trans. Graph. 33, 4, Article 38 (July 2014), 11 pages. DOI = 10.1145/2601097.2601184 http://doi.acm.org/10.1145/2601097.2601184.

0730-0301/14/07-ART38 $15.00.
DOI: http://dx.doi.org/10.1145/2601097.2601184

1 Introduction

Numerical wave simulation generates environmental sound effects of compelling realism that complement visual effects, reveal information about occluded parts of the scene, and establish a scene-dependent mood. But it is too expensive to compute in real time. Precomputed approaches allow real-time performance for dynamic sources in static scenes by storing the simulated impulse response (IR) as a function of source and listener position. Memory cost is prohibitive, requiring hundreds of megabytes even for a constrained 2D distribution of source locations in small scenes [Raghuvanshi et al. 2010]. To render the acoustics at run-time, a convolution is performed on the anechoic (unpropagated) audio signal emitted by each source with the IR between the source and receiver locations. This expensive processing is typically performed on a busy shared audio core, allowing only a few dynamic sources.

We take a novel parametric approach to more compactly encode the wave field and render it faster. The acoustic field in a typical environment is spatially chaotic, making it hard to compress directly [Ajdler et al. 2006]. On the other hand, our perception of that environment changes smoothly as we or the sound source moves – the human auditory system extracts only a few salient properties such as loudness, directionality, and reverberance, not individual information about the huge number of diffracted and scattered wavefront arrivals [Gade 2007]. We therefore convert a field of IR signals into a set of scalar fields corresponding to a few perceptual parameters (Figure 1). We observe that even in scenes of high geometric complexity, each resulting parameter field is smooth as expected, which we exploit using lossless compression similar to PNG. Our approach reduces memory by orders of magnitude compared to [Raghuvanshi et al. 2010]. Scaling results (Section 7) show that spatial compression essentially removes an entire spatial dimension from the 6D field over 3D source and listener locations.

Our run-time engine reverses the encoding to apply IRs that conform to the precomputed parameters. We capitalize on the considerable flexibility involved using a novel, fast rendering algorithm. It uses a small set of *canonical filters*, optimized to cover the parameter space, allow artifact-free linear interpolation, and yield physically plausible results. Instead of interpolating and convolving an IR for each source, we split each source signal into scaled copies and accumulate a sum over all sources for each canonical filter. This fixed-bus architecture produces identical results but largely re-

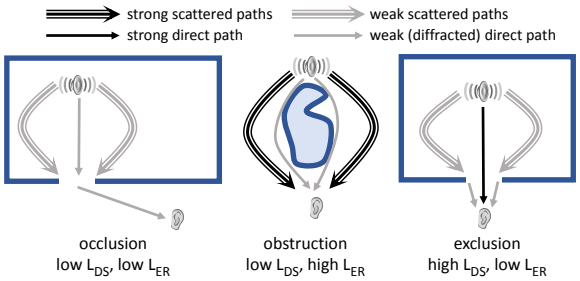


Figure 2: Path-dependent propagation effects.

moves the the computation's dependence on the number of sources, providing a large speedup.

It is usual in acoustic analysis to segment an IR into three transient phases: *direct sound* (DS), *early reflections* (ER), and *late reverberation* (LR) [Kuttruff 2000]. We find that much of the perceptual quality of point-to-point sound propagation in an arbitrary scene can be captured by just four parameters: DS and ER loudness, and decay times of the ER and LR. Refer to Figure 3. This parameter set is by no means complete and neglects important propagation effects including accurate variation in sound directionality (spatialization), diffusion, and coloration. Nevertheless, it provides immersive auralization supporting a wide range of effects. Figure 2 shows three, called *occlusion*, *obstruction*, and *exclusion*.

These effects integrate a huge number of sound paths emanating in all directions from the source, diffracting around obstacles, and scattering many times off scene geometry before arriving at the listener. They are notoriously difficult to specify by hand or compute on the fly, and so are ignored or replaced by unrealistic heuristics in many current games. The absolute and relative loudness of direct and reverberant energy also provides an important distance cue [Kolarik et al. 2011], currently specified by hand in game audio design. All these effects are elegantly captured by just two loudness parameters. Two additional decay rate parameters indicate the reflectivity, size and openness of the space connecting source and listener.

As discussed in Section 2, our parameters are related to ones developed in earlier work. In room acoustics, parameters are extracted to guide architectural design and analysis, not for auralization. In interactive audio for games, reverberation parameters are manually specified by artists and augmented with non-physical heuristics. For example, parameters are selected based on which hand-marked room the listener occupies. Occlusion is driven using one or a few ray visibility queries. Sound is forced to attenuate to 0 at an artist-specified radius to prevent it from propagating through walls.

We introduce the first parametric field approach for interactive rendering of wave acoustics, and show that its results are easy to compress and interpolate, and fast to render. It handles much larger scenes than possible with previous techniques: an entire level of a modern game now fits in a 100MB memory budget. We introduce a robust encoder to extract acoustic parameters from band-limited wave simulations which works for billions of IRs. We also describe a novel, high-performance run-time rendering technique. We demonstrate our system's quality and performance by integrating it with a modern game engine.

2 Related Work

Interactive geometric acoustics (GA) Takala and Hahn introduced interactive acoustics to CG using ray tracing [1992]. Beam tracing [Funkhouser et al. 2004] supports real-time walkthroughs of static scenes with approximate diffraction from long straight edges (>1m). Accelerated beam-tracing [Laine et al. 2009] can handle a moving source. Frustum tracing [Chandak et al. 2008] computes a fast conservative approximation to beam tracing using existing fast ray tracers and supports dynamic sources.

The image source method (ISM) represents the global field as a superposition of expanding spherical wavefronts centered at image sources [Allen and Berkley 1979]. Their locations are computed by recursively reflecting the actual source location through all planes in the scene and retaining ones with clear line-of-sight to the listener. ISM is suitable for modeling specular reflections from large (meters across) flat reflectors, not for complex, scattering geometry. The number of image sources grows exponentially in the number of reflections. Beam/frustum tracing can be seen as implementations of ISM (each beam represents an angular section of a wavefront) thus sharing these limitations. [Tsingos 2009] proposes a faster but more approximate ISM-based technique that coarsely samples sources (one per room) and computes a set of image sources for each, interpolating their locations when sources move. Scattering and diffraction are ignored.

Recent systems [Taylor et al. 2009; Schröder 2011] combine such ISM techniques for specular reflections with Monte-Carlo ray tracing for diffuse scattering on simplified scene models. Automatic simplification remains a challenging problem [Siltanen 2005]. Diffraction can be modeled using the Biot-Tolstoy-Medwin (BTM) diffraction theory [Svensson et al. 1999]. When a ray hits an edge element, [Svensson et al. 1999] provides analytic directivity functions for scattered rays in all directions. A probabilistic variant is better suited for energy-based methods which neglect phase [Stephenson and Svensson 2007].

All GA techniques use a Lagrangian model that propagates pressure/energy quanta along piecewise straight paths. Computation increases exponentially as the number of reflection, scattering, and diffraction events along the path (*path order*) increases. Interactive systems limit the path order. Accurately modeling high-order diffraction and scattering is an active research area [Calamia 2009; Siltanen et al. 2009].

Offline solvers Time-domain wave solvers use an Eulerian model of a time-stepped pressure field, incurring an expense that increases with the scene's volume and the maximum frequency but insensitive to its geometric complexity or the order of wave-geometry interactions. Such solvers have been applied to room acoustics [Savioja et al. 1994; Murphy et al. 2007; Kowalczyk and van Walstijn 2010]. Modern hardware and algorithms make it feasible to simulate offline up to mid-frequencies (above 1kHz) in concert-hall-sized scenes [Sakamoto et al. 2008; Mehra et al. 2012], or real-time in rooms [Savioja 2010].

Wave simulation accurately models diffraction and scattering and so is excellent for precomputing acoustics. Simulation accuracy is essential; salient propagation characteristics like total loudness depend on complex, multi-path physics. Propagated sound often integrates over a huge number of weak high-order paths. An example is our Sanctuary scene, where a main hall opens outdoors through a cave-like passage. Sound has to reflect, scatter, and diffract many times within the hall, through the cave, and out the cave's mouth to reach a listener standing to the side. Such cases are extremely challenging for GA methods.

[Siltanen et al. 2007] unify GA with a path-integral formulation analogous to Kajiya's rendering equation in graphics, and propose acoustic radiance transfer (ART), an offline technique inspired by progressive radiosity. Arbitrary BRDFs are supported but highly specular reflections are costly. Diffraction can also be approximately modeled [Siltanen et al. 2010b]. Hybrid solvers [Southern et al. 2013] combine wave simulation at low frequencies with ART for diffusion/late reverberation and beam tracing for specular bounces. Such ideas are promising for extending our wave-based

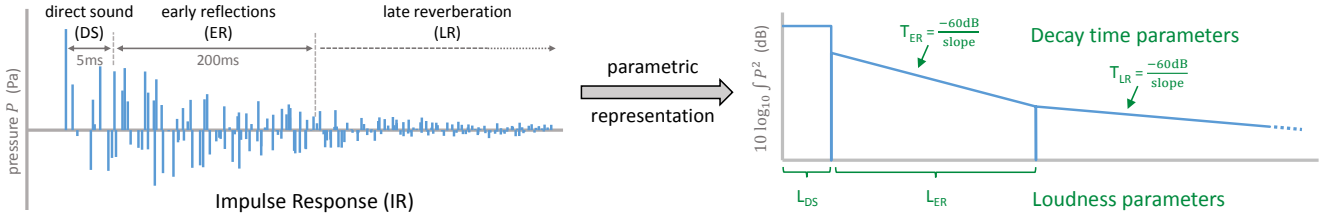


Figure 3: Parametric IR encoding schematic (time not to scale). The four parameters we extract are shown in green on the right for an IR shown on the left.

precomputation to higher frequencies. Geometric and numerical wave techniques are compared in [Siltanen et al. 2010a].

Real-time wave acoustics Prior work proposes wave-based precomputation with real-time auralization [Raghuvanshi et al. 2010]. Our technique reduces memory by orders of magnitude (see Section 7), and accelerates the run-time. It also allows a true 3D (rather than 2D) sampling of source positions to support a flying source/listener. More recent work on large, outdoor spaces [Mehra et al. 2013; Yeh et al. 2013] requires that either the listener or the sources be static. These techniques also require manual partitioning of the scene into well-separated objects.

Acoustic parameters Sabine's pioneering work quantified the subjective quality of a concert hall using the simple measure of reverberation time [1953]. Since then, one of the primary goals of room acoustics research has been to find a set of parameters derived from physical IRs that completely capture the sound quality of a performance space [Krokstad 2008; Beranek 2003]. The widely-used standard [ISO 3382-1:2009] codifies a small set of parameters and prescribes how they can be extracted from measured IRs. Sampling these parameters at a few (~10) listener locations lets an acoustician decide if a hall's acoustics are suitable and make architectural changes. The source distribution is fixed, corresponding to musical instruments on the stage.

We are instead interested in recreating dense sound fields in general 3D scenes, between arbitrary, moving pairs of points possibly lying in different rooms, across obstacles, or even outdoors. Several new problems must be solved: 1) designing automatic techniques for extracting parameters from the time- and frequency-limited IRs resulting from simulation, 2) minimizing storage for dense parametric fields and exploiting their spatial compressibility, and 3) auralizing these parameters convincingly with high performance.

Parametric sound fields have been proposed before [Stettner and Greenberg 1989] and are supported by commercial software [Rindel and Christensen 2013]. In both cases, the simulation is based on ray tracing and the goal is acoustic visualization for design, not efficient encoding and generation of an acoustic experience.

Complementary to this work in room acoustics, consumer audio applications have used parametric reverberation filters to manually design a wide range of propagation effects. The I3DL2 standard [IAS 1999] codifies a set of parameters using the vocabulary of occlusion/obstruction/exclusion effects mentioned in the introduction. ISO 3382-1 and I3DL2 share parameters for separate levels of direct, reflected and reverberant sound. ISO 3382-1 omits LR decay time; I3DL2 omits ER decay time. Other parameters in ISO 3382-1 (lateral energy levels) presume a canonical view direction looking towards the stage. While I3DL2 generalizes the scene configuration, acoustic calculation or parameter extraction is outside its scope; audio designers usually specify the parameters by hand or using simple heuristics. In contrast, our system captures how acoustic parameters vary with source and listener location, from numerical wave simulations.

3 Precomputed Sound Simulation

The input to our system is the scene geometry represented as a "triangle soup" with associated materials, supporting typical game maps. Scene triangles are voxelized into a 3D occupancy grid for simulation, along with their material codes. The maximum desired simulation frequency, $\nu_{\max}$, determines the cell size Δ. The decision is based on memory and computational constraints. We use the ARD solver [Raghuvanshi et al. 2009] which determines voxel size via $\Delta = {}^3/_8\,\lambda_{\min}$, where $\lambda_{\min} = c/\nu_{\max}$ is the minimum wavelength and c is the speed of sound. This represents 2.7 samples per wavelength. We typically fix $\nu_{\max} = 500\text{Hz}$ yielding $\Delta = 25.5\text{cm}$. Finite difference simulation could also be used, requiring about 5 samples per wavelength.

The entire 7D pressure field is denoted $P(\mathbf{x_s}, \mathbf{x}_\ell, t)$, where $\mathbf{x_s}$ and $\mathbf{x}_\ell$ are the source and listener locations and t is time. We uniformly discretize the 6D space spanned by $(\mathbf{x_s}, \mathbf{x}_\ell)$. Simulations are performed independently over a 3D set of *probe* source locations $\mathcal{X}_s = \{\mathbf{x_s}\}$. At each $\mathbf{x_s}$, a wave simulation yields a 4D slice through this 7D field with $\mathbf{x_s}$ held fixed, denoted $P_{\mathbf{x_s}}(\mathbf{x}_\ell, t)$.

Source probe sampling Probe sources are placed in the scene with user-specified uniform spacing in the horizontal and vertical directions. Vertical spacing at roughly human height ensures that entire building floors, doorways, etc. are not missed. Horizontal spacing is typically 2-4m and vertical spacing 1.6m. Optionally, watertight *region-of-interest* meshes can be specified which constrain probe samples to their interior. These are a few (2-42) boxes in our examples. They are designed to include all places the listener might walk or fly, and exclude unwanted regions such as below the ground. Region-of-interest meshes are voxelized to compute a discrete union of their interiors. Any probe sample whose corresponding voxel lies outside the region of interest or within scene geometry is rejected. This generates $\mathcal{X}_s$. Note that we use listener navigation to guide source probe sampling because source and listener locations are swapped at run-time as explained in Section 5.1.

Per-probe simulation At each probe $\mathbf{x_s} \in \mathcal{X}_s$, we simulate inside a vertical cylinder around it of a specified radius and top and bottom heights. Our rationale is that sounds do not propagate arbitrarily far but attenuate due to occlusion/absorption and with distance. Typical values we use are a 45m radius (50m after padding, described shortly) and 15-20m height. This is roughly the diameter of a city block and the height of 4-5 building floors. We note that propagation by 50m results in a pure distance attenuation of -34dB relative to the loudness at 1m.

A padding layer of air is added around this cylindrical region, which aids in run-time extrapolation as will be described later. Its thickness is kept larger than the listener sample spacing used during encoding. The entire outer surface of the simulation region is marked as "fully absorbing" to model emission into a free field.

Within this cylindrical domain, we invoke the wave simulator. Its duration is $t_{\max} = \Delta_{DS} + \Delta_{ER} + \Delta_{LR} + \Delta_C$ as defined in Table 1, where Δ_C is the line-of-sight delay from $\mathbf{x_s}$ to the furthest point

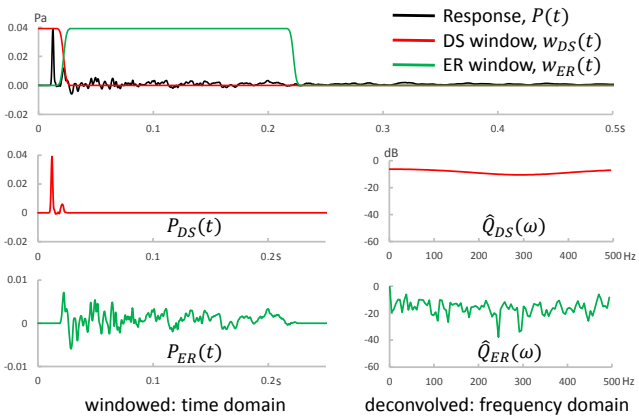


Figure 4: Encoder extraction of loudness parameters. The response $P(t)$ here is an actual output from our simulator.

on the cylinder's surface. This ensures that every point receives a sufficiently long signal to encode. In our simulations, $t_{\max} \approx 1$s. A source pulse is introduced at location $\mathbf{x_s}$ via

$$\tilde{s}(\mathbf{x}, t) = \gamma\, \delta(\mathbf{x}, \mathbf{x_s})\, s(t),\ s(t) = \exp\left(\frac{-(t-t_0)^2}{\sigma_s^2}\right), \quad (1)$$

where δ denotes the 3D Kronecker delta function, $\mathbf{x}$ and $\mathbf{x_s}$ are drawn from the discrete set of voxel centers, $\sigma_s = \sqrt{\ln 10}/(\pi\, \nu_{\max})$, and $t_0 = 5\,\sigma_s$. The signal $\tilde{s}$ is a Gaussian introduced at a single cell. The initial delay t_0 ensures a very small starting amplitude (< -210dB relative to peak). The factor γ normalizes the signal to have unit peak amplitude at a distance of 1m. For our solver (ARD), $\gamma = 1/(0.4\,\Delta)$. The choice of σ_s forces the Gaussian's spectrum to decay by -20dB at frequency $\nu_{\max}$. This limits aliasing but still contains extractable information near $\nu_{\max}$.

Subsequent processing The entire simulation field is stored on disk and provided to the encoder, which does a streaming read of the data in blocks from the disk, encodes each block, and then writes out the corresponding 3D block for the output parameter field. This simulate+encode process can be done for all probe sources in a massively parallel computation. The four 3D parameter fields over all probes are concatenated to be loaded by the run-time system.

4 Encoder

We separately encode each 4D slice of the wave field with $\mathbf{x_s}$ held fixed, as produced by a single wave simulation. The first step is parameter extraction, denoted

$$P_{\mathbf{x_s}}(\mathbf{x}_\ell, t) \longmapsto \{F^{param}_{\mathbf{x_s}}(\mathbf{x}_\ell)\},$$

where $param \in \{L_{DS}, L_{ER}, T_{ER}, T_{LR}\}$. Figure 1 shows examples of such fields. These fields are then smoothed and spatially sampled, and finally quantized, compressed and stored. We describe these steps in more detail in the following. The Fourier transformation of a signal $s(t)$ is denoted $\hat{s}(\omega)$.

4.1 Parameter Extraction

Processing is performed independently on the response signal received at each listener cell $\mathbf{x}_\ell$ and relies on concepts from room acoustics [Gade 2007]. Recall from Section 3 that $P_{\mathbf{x_s}}(\mathbf{x}_\ell, t)$ is not a true impulse response but rather the response to an omnidirectional Gaussian pulse, $\tilde{s}(\mathbf{x}, t)$, emanating from $\mathbf{x_s}$. We simplify notation in this section and write $P_{\mathbf{x_s}}(\mathbf{x}_\ell, t)$ as $P(t)$. Table 1 lists the processing parameters that govern parameter extraction.

symbol	meaning	value
D_τ	threshold for first arrival	-90dB
D_{ER}	falloff indicating start of ER decay	-3dB
Γ_w	partitioning window width factor	3
Δ_{DS}	DS duration	5ms
Δ_{ER}	ER duration	200ms
Δ_{LR}	duration for LR slope estimation	600ms

Table 1: Encoder processing constants.

Propagation delay In room acoustics, the direct path is nearly unoccluded in concert halls and the direct energy can usually be analytically estimated and removed. Our encoding must capture more complex, scene-dependent occlusion. There is an initial delay before energy starts arriving which we denote $\tau(\mathbf{x_s}, \mathbf{x}_\ell)$, as the initial wavefront emanating from $\mathbf{x_s}$ propagates outward at speed c before reaching the listener at $\mathbf{x}_\ell$. This delay corresponds to a geodesic path that may diffract around scene geometry and become extremely attenuated. We adopt a simple definition viz.

$$\tau(\mathbf{x_s}, \mathbf{x}_\ell) = \min_t\{10\, \log_{10} P^2(\mathbf{x_s}, \mathbf{x}_\ell, t) > D_\tau\},$$

where the threshold D_τ indicates absence of any significant response. A value of -90dB works robustly in all our examples. D_τ must be chosen with care. Too large a value misses a weak initial response in a highly occluded situation. Too small a value is triggered by numerical noise, which travels faster than sound in spectral solvers like ours. We can typically vary this value by about 10dB without substantial impact on τ.

Subsequent steps rely on τ but we do not retain this parameter after processing. Propagation delay is typically neglected in games because players can mistake audio-visual asynchrony for system latency in the audio pipeline.

Loudness While the term "direct sound" is standard in acoustics, a more appropriate term is "first arriving sound" since its path may be indirect and its perceptual loudness may integrate other reflected/scattered paths that arrive within a few milliseconds of the shortest path. It is known that our hearing system combines all such information into a single perception [Gade 2007]. We assume the interval $t \in [\tau, \tau + \Delta_{DS}]$ contains the initial sound with $\Delta_{DS} = 5$ms. The extraction process is summarized in Figure 4.

Separating this interval in $P(t)$ must be done carefully using a smooth windowing function. Using a step function in the time domain results in Gibbs ripples which contaminate the spectral processing we do later. We use the Gauss error function, defined as

$$w(t) = \frac{1}{\pi\sigma_w} \int_{-\infty}^{t} \exp\left(\frac{-t'^2}{\sigma_w^2}\right) dt'.$$

We fix $\sigma_w = \Gamma_w\, \sigma_s$ where σ_s is the Gaussian source signal's standard deviation from Eq. 1. The proportionality constant Γ_w controls the smoothness of the partitioning; we set $\Gamma_w = 3$. The error function grows monotonically from 0 to 1 without oscillation, is controllably compact in time, and provides a straightforward partition-of-unity $w(t) + w(-t) = 1$. The complementary window is denoted $w'(t) = w(-t)$.

To estimate direct sound loudness, we first extract the segment $P_{DS}(t) = P(t)\, w_{DS}(t)$, where $t \in [\tau, \tau + \Delta_{DS} + 4\sigma_w]$ and the time window is $w_{DS}(t) = w'(t - \tau - \Delta_{DS} - 2\sigma_w)$, shown in Fig. 4 (second row). Next, we transform the signal to the frequency domain to obtain $\hat{P}_{DS}(\omega)$ and deconvolve it with the source signal to obtain the underlying frequency response $\hat{Q}_{DS}(\omega) = \hat{P}_{DS}(\omega)/\hat{s}(\omega)$. Finally, we compute the en-

ergy over bands between the set of n_ν octave frequencies (in Hz) $\nu_i = \{62.5, 125, 250, 500, \ldots, \nu_{\max}\}$, via

$$\left\{L_i^{DS}\right\} = 10\,\log_{10}\left(\frac{1}{\nu_{i+1}-\nu_i}\int_{\nu_i}^{\nu_{i+1}}\left\|\hat{Q}_{DS}(2\pi\nu)\right\|^2 d\nu\right). \quad (2)$$

Direct sound loudness averages these:

$$F^{L_{DS}} = \frac{1}{n_\nu}\sum_i L_i^{DS}.$$

Note that we do not deconvolve the entire input signal to convert a Gaussian response to an IR, but instead first window in time and then deconvolve in the frequency domain where energy can be estimated directly via Parseval's theorem. This avoids Gibbs ringing that arises when deconvolving a band-limited response and can overwhelm its important properties especially when the direct pulse has high amplitude (i.e. when $\mathbf{x}_\ell$ is close to $\mathbf{x_s}$).

Loudness parameter extraction for the ER is similar to that for the DS phase, as shown in Fig. 4 (bottom row). The ER interval is extracted from the response $P(t)$ via $P_{ER}(t) = P(t)\,w_{ER}(t)$ where $t \in [\tau + \Delta_{DS},\, \tau + \Delta_{DS} + \Delta_{ER} + 4\sigma_w]$ and $w_{ER}(t) = w(t-\tau-\Delta_{DS}-2\sigma_w)\,w'(t-\tau-\Delta_{DS}-\Delta_{ER}-2\sigma_w)$. Energy is then extracted as described above for direct sound.

Decay times The energy profile of reverberation in typical rooms decays differently as time progresses. Steeply decaying initial reflections are often followed by a more slowly decaying LR. The *early decay time* (EDT, denoted T_{ER}) and *late reverberation time* (LRT, denoted T_{LR}) parameters account for separate energy decay rates in these two phases. It is known that EDT strongly depends on the scene geometry and on the locations $(\mathbf{x_s}, \mathbf{x}_\ell)$, while LRT depends mainly on the scene's volume and surface area. It is also known that for impulsive sources such as a gunshot or clap, LRT correlates well with subjective reverberance, while for continuous sources such as speech and music, EDT is a better measure [Gade 2007]. We extract and encode both.

Decay times are estimated using the backward (Schroeder) integral, defined as $\tilde{I}(t) = \int_t^\infty P^2(t)\,dt$. The estimation is usually done in a single octave band; we employ the 250-500Hz band for which the response first needs to be band-passed. Band-passing the entire response works poorly, introducing ringing that contaminates the weak signal near its end. We apply a standard but more compute-intensive approach based on the short-time Fourier transform (STFT), described in the following.

The direct sound typically causes a jump in the energy decay curve which we remove by time-windowing it out:

$$P_{\neg DS}(t) = P(t)\,w(t-\tau-\Delta_{DS}-2\sigma_w). \quad (3)$$

For spectral analysis, we employ a sliding Hamming window of width 87ms (256 samples for $\nu_{\max} = 500$Hz) and 75% overlap. For each translation of the window starting at time τ, we multiply the window against $P_{\neg DS}$ and compute the FFT of the windowed segment. Then we take the spectrum's sum of squared magnitudes in the 250-500Hz band, yielding the energy $E(t)$.

The energy decay curve applies the Schroeder integral to obtain

$$I(t) = 10\,\log_{10}\int_t^{t_{\max}} E(t)\,dt, \quad (4)$$

where $t_{\max}$ denotes the termination time of the simulated response P. Combining time-windowed STFT and integration yields a smooth curve. Figure 5 shows an example.

EDT estimates the **initial** decay time of I. [ISO 3382-1:2009] (hereafter referred to as "the ISO") recommends linear regression

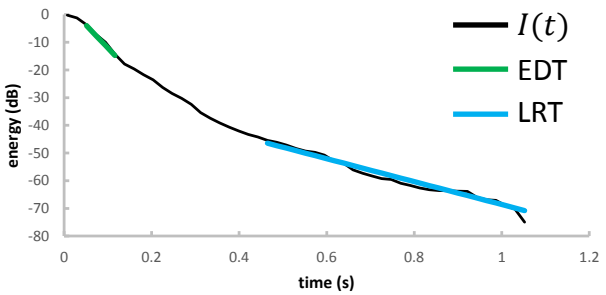


Figure 5: Encoder extraction of decay time parameters. The energy decay curve, $I(t)$, is defined in Eq. 4, and represents an actual result from our system on a simulated pressure response.

on the first 10dB of the decay. Since we have removed the direct sound in Eq. 3, $I(t)$ contains a plateau near $t = 0$ not present in the true decay. We thus ignore the initial part of I until it decays by $D_{ER} = -3$dB and call this time t_0. Next, following the ISO, we find the time when the signal has decayed a further 10dB relative to D_{ER}, called t_1. To estimate slope, δ_{ER}, in the interval $[t_0, t_1]$ we depart somewhat from the ISO recommendation. Real decay curves are often concave, especially outdoors. Linear regression underestimates the decay rate in such cases and so overestimates EDT. We compute derivatives on the smooth I using forward differencing to obtain a time-varying slope. The RMS of these yields the EDT slope, and puts more weight on an initial fast decay. Finally, $T_{ER} = -60/\delta_{\mathrm{ER}}$ yields the time necessary for the energy to decay by 60dB, assuming the decay rate continues as estimated. When the decay really is linear, our approach matches the ISO.

LRT estimates the **asymptotic** decay time for I. We consider its end segment $t \in [t_{\max} - \Delta_{LR}, t_{\max}]$, use linear regression to find the slope, and set $T_{LR} = -60/\delta_{LR}$.

4.2 Parameter Field Preparation

Direct sound loudness (L_{DS}) exhibits a singularity at the probe location, $\mathbf{x_s}$ due to distance attenuation, so we encode it relative to the free-field attenuation of a monopole source. That is, we update via $F_{\mathbf{x_s}}^{L_{DS}}(\mathbf{x}_\ell) \leftarrow 20\log_{10}\|\mathbf{x}_\ell - \mathbf{x_s}\| + F_{\mathbf{x_s}}^{L_{DS}}(\mathbf{x}_\ell)$. This improves compression and reduces the dynamic range. (Our parameter field visualizations show the uncompensated field to make its behavior more intuitive.) Loudness parameters are encoded in logarithmic space (Eq. 2) clamped to the range -70dB to +20dB. The upper bound is conservative since passive acoustic amplification due to wall reflections rarely exceeds +6dB. The lower bound corresponds to the audibility of a source with loudness 80dB SPL at 1 meter (which damages hearing on continuous exposure) decaying to barely audible at 10dB SPL. Decay time parameters (T_{ER} and T_{LR}) are encoded in logarithmic space as well, via $\log T_{ER}/\log 1.05$. The denominator ensures 5% relative increase between consecutive integral values. We use a range of -64 to 63, representing 44ms to 21.6s. Each parameter is clamped against its bounds. Parametric coding allows the parameters to be coarsely sampled in space with little aliasing: smoothing and subsampling of the parameter fields is performed using a box filter over all simulated samples. A sample lying inside scene geometry ("bulkhead") is given a special don't-care code used during compression.

4.3 Quantization

According to the ISO, the perceptual just-noticeable-difference (JND) is 1dB for loudness and 5% relative for decay times, under critical listening conditions. Each quantum in our logarithmic mapping for both loudness and decay time parameters corresponds

to one JND. With these ranges and quantization steps, each mapped scalar parameter fits in a byte. This quantization level is unduly conservative for interactive applications where conditions are not ideal and multiple sources play simultaneously. Loudness fields are known to exhibit spatial oscillations around 1dB [Bradley and Halliwell 1988]; late reverberation time also exhibits spatial fluctuations [Davy et al. 1979]. We therefore increase the quantization threshold (Δq in the next section) from 1 to 3 integral steps (that is, 3dB for loudness and 15% relative increment for decay times), to gain a factor of two improvement in compression. Even with an increased quantization threshold, a field oscillating just around a quantization level continually switches between discrete values, reducing compression. We avoid this problem using the differencing process described below.

4.4 Compression

The four parameter fields are treated identically, as 3D arrays with a bulkhead code indicating presence of geometry. Each 2D Z slice is compressed separately, where Z points up, against gravity. Thus, only a few slices must be decompressed at run-time if the user persists at roughly the same height while moving through the scene.

Our technique is a subset of PNG optimized to deal with bulkheads and accelerate decompression. We consider each X scanline in turn, accumulating a residual, r, representing the as-yet unquantized running difference. This residual always stays below the quantum, denoted Δq. We also maintain the previously processed field value f' and the current field value f, and subtract to yield the running difference $\Delta f = f - f'$. Initially, $f' = r = 0$. We compute the output, q, and update the residual via

$$q \leftarrow \left\lfloor \frac{\Delta f + r}{\Delta q} \right\rfloor, \quad r \leftarrow r + \Delta f - q\,\Delta q. \tag{5}$$

This simple procedure uses the scanline's previous value as the predictor. If a bulkhead is encountered, we set $f = f'$ producing the value $q = 0$ over its span. A transition cost is incurred only when the scanline exits the bulkhead. We finally perform LZW compression using Zlib over the resulting stream of q values.

5 Run-time

5.1 Parameter Decoding

As described in Section 3, the precomputed output concatenates data over a set of probe source locations $\mathcal{X}_s$. When this encoding is loaded, probe locations are inserted into a spatial data structure; we use a simple grid. Given a continuous source location at runtime, this data structure accelerates the lookup of 8 probes forming a box around it. Some of these may be missing because they lie inside walls or outside the specified region of interest. We further remove all probes that are "invisible" to the run-time source to avoid interpolating across walls, using a finely-sampled voxelization of the scene. This results in a valid set (≤ 8) of probes whose tri-linear weights are renormalized. The final parameter values are obtained via a weighted sum of each parameter value at the listener location.

Computing the parameter value at the listener again involves trilinear interpolation. Each probe comprises a 3D parameter field organized as a set of compressed Z slices. The two slices spanning the listener location are decoded via LZW-decompression and de-quantized by reversing Eq. 5 to obtain a 2D array for each parameter. We then interpolate over the 8-sample box around the listener. Invalid samples are removed in the same way as for probe sources and the weights renormalized. This yields the (sampled) probe's parameters at the continuous listener location. The entire process represents 6D hypercube interpolation. The computation is extremely fast. The costliest part is LZW slice decompression. We accelerate it by storing decompressed Z-slices in a 1MB global cache with a least-recently-used policy.

Speedup from acoustic reciprocity We obtain a 10-100× performance gain by exploiting acoustic reciprocity. The IR between a point source and listener remains the same if these locations are interchanged; the same holds for acoustic parameters. At run-time, we simply exchange the source and listener location and apply the procedure described above. The player becomes the source, and the problem is converted from multiple-source single-listener to multiple-listener single-source. The latter is faster because it reduces the number of fields to be decoded to at most 8 probes, rather than 8 times the number of sources. Since LZW decompression is the bottleneck, the performance gain is substantial.

Extrapolation outside the simulation region If the evaluation point lies outside the simulation cylinder for the probe source, we extrapolate a free-field extension. Recall from Section 3 that we apply an air padding around the simulation cylinder. This guarantees that cells on its surface avoid scene geometry and approximate radiation into a free field. We find the cell where the ray from the probe to the evaluation point exits the cylinder and look up the parameter values there. This heuristic works well for sounds propagating outdoors from open doors or over low walls. It does not account for distant geometry that scatters or blocks significant energy, which must be included in the simulation cylinder during precomputation.

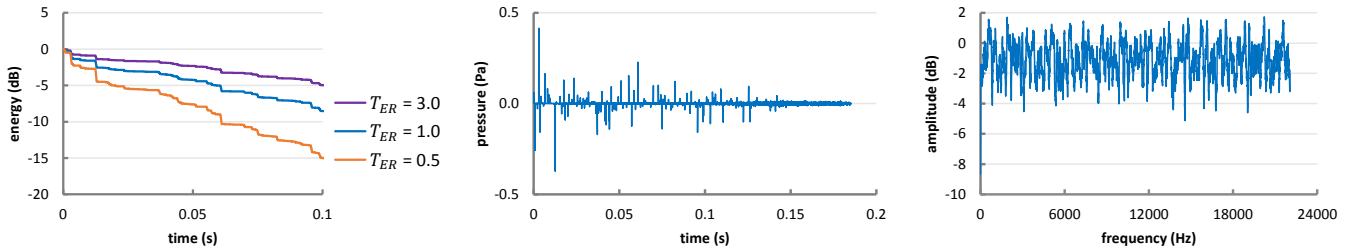
5.2 Parameter Rendering

Our acoustic rendering method uses a small, fixed number of global canonical filters (CFs) whose output achieves the effect of applying individual IRs. For each source-listener pair, the acoustic filter we apply at run-time must reproduce the properties represented by its four parameter values. We exploit the freedom available in choosing its exact form. Consider the i^{th} source with monaural signal $s_i(t)$ (distinct from the preprocessed source signal $s(t)$), for which we obtain $\{L_{DS}, L_{ER}, T_{ER}, T_{LR}\}$ for the source and listener location, as above. We apply a stereo (binaural) filter $h_i(t)$ which respects the four parameters, producing as stereo output $o_i(t) = s_i \star h_i$, where $\star$ denotes stereo convolution (s_i is input to both filter channels). h_i can be broken into three temporally contiguous parts as $h_i = h_i^{DS} + h_i^{ER} + h_i^{LR}$. Rendering can thus be expressed as a sum of three convolutions: $o_i(t) = o_i^{DS} + o_i^{ER} + o_i^{LR} = h_i^{DS} \star s_i + h_i^{ER} \star s_i + h_i^{LR} \star s_i$.

Direct sound The direct sound filter, h_i^{DS} needs to scale s_i by the encoded loudness L_{DS}. Since distance attenuation is removed during encoding (Section 4.2), it must also be applied. The net scale factor is $10^{L_{DS}/20}/d$, where d is source-to-listener distance. Spatialization is performed based on the source location and the listener's location and orientation. Game audio engines already provide these as low-latency operations, producing stereo output for the direct sound, $o^{DS}(t)$.

The other two filters, h_i^{ER} and h_i^{LR}, must together respect three parameters, $\{L_{ER}, T_{ER}, T_{LR}\}$, representing a loudness and two slopes (dB/s), and satisfy the constraint that the energy decay of $h_i^{ER} + h_i^{LR}$ be continuous where its components meet in time.

Early reflections We express the ER filter in terms of $n_{ER} = 3$ canonical filters $\left\{H_j^{ER}\right\}$. As described in Section 6, the set of CFs are designed to have unit energy that decays exponentially with the specified EDT T_{ER}^j. Note that we index source sounds by i and canonical filters by j. Given T_{ER}, we interpolate over the brack-



(a) energy decay profiles (b) canonical IR (T_{ER} = 1.0), time domain (c) canonical IR (T_{ER} = 1.0), frequency domain

Figure 6: Canonical filters for the ER phase. The three filters satisfy the energy decay curves shown in (a). The left stereo channel for one of the three ($T_{ER} = 1.0s$) is shown on the right, in both the time (b) and frequency (c) domains. Note the decay of its specular part and buildup of its diffuse part in the time domain, and overall flat (colorless) frequency response.

eting CFs for which $T_{ER}^j \leq T_{ER} \leq T_{ER}^{j+1}$. We use EDT values of 0.5, 1.0 and 3.0s; any T_{ER} outside this range is clamped. We find the interpolation weights, α_j^{ER} and α_{j+1}^{ER}, by assuming that all decay curves are perfectly exponential and requiring that the linearly interpolated result match the ideal exponential decay for T_{ER} at a "matching time", t^*. We choose the middle of the ER: $t^* = \Delta_{ER}/2 = 100\text{ms}$. This choice ensures that the interpolated filter's early decay time has a maximum relative error less than 5%, comparable to a JND as per the ISO. Finally, the weights are multiplied by a factor to enforce loudness, yielding

$$\alpha_j^{ER} = 10^{-L_{ER}/20} \frac{10^{-3\,t^*/T_{ER}^{j+1}} - 10^{-3\,t^*/T_{ER}}}{10^{-3\,t^*/T_{ER}^{j+1}} - 10^{-3\,t^*/T_{ER}^{j}}}, \tag{6a}$$

$$\alpha_{j+1}^{ER} = 10^{-L_{ER}/20} - \alpha_j^{ER}. \tag{6b}$$

We approximate h_i^{ER} as a linear combination of canonical filters,

$$h_i^{ER} \approx \sum_{j=1}^{n_{ER}} \alpha_j^{ER}\left(L_{ER}^i, T_{ER}^i\right)\, H_j^{ER}(t). \tag{7}$$

Substituting this into the expression for rendered output $o^{ER}(t) = \sum_i h_i^{ER} \star s_i$, and interchanging summations, we get

$$o^{ER}(t) \approx \sum_{j=1}^{n_{ER}} H_j^{ER} \star \sum_i \alpha_j^{ER}\left(L_{ER}^i, T_{ER}^i\right)\, s_i. \tag{8}$$

This is the main idea of our run-time: express the filtering for all sources as a sum of outputs of fixed filters applied to linear combinations of the source signals, with weights determined by (dynamically varying) per-source acoustic parameters.

Late reverberation Processing to compute the LR output is similar. We use $n_{LR} = 3$ CFs with decay times of 0.75, 1.5 and 3.0s, and define the matching time as $t^* = 0.75\,T_{LR}$. These choices yield relative error less than 5.7%, again comparable to the JND. Unlike the case of the ER, loudness for the LR, which we denote L_{LR}, is not stored explicitly but instead derived by enforcing energy continuity at the end of the ER. We estimate energy in the 25ms tail of the interpolated filter h_i^{ER} calculated above, given by the linear combination $h^{ER} = \sum_j \alpha_j^{ER} H_j^{ER}$. Denote as E_j^* the (precomputed) total energy in the last 25ms of H_j^{ER}, averaged over both stereo channels. We estimate $E^* = \left(\sum_j \alpha_j^{ER}\sqrt{E_j^*}\right)^2$, from which $L_{LR} = 10\log_{10} E^*$. Now, analogous to Eq. 6, we use L_{LR} and T_{LR} to find the LR interpolation coefficients, α_j^{LR} and α_{j+1}^{LR}.

Optimized convolution We use partitioned convolution to apply CFs to the weighted sum of source signals as required by Eq. 8. Since the filters are fixed, we pre-transform them to the frequency domain to avoid costly run-time FFTs. Increasing the partition size n^* reduces the per-sample computational cost but increases latency because a partition must be full for frequency-domain convolution to be performed. Fortunately, our filters include inherent delay with respect to the direct sound: the first Δ_{DS} seconds of H_j^{ER} and $\Delta_{DS} + \Delta_{ER}$ seconds of H_j^{LR} are zero. Choosing a partition size equal to this delay and time-shifting the filter to remove its initial zeroes increases performance while preserving the desired output. A power-of-two partition size is desirable for FFTs. We choose $n^* = 512$ samples for H_j^{ER} and $n^* = 8192$ samples for H_j^{LR}. At 44100Hz, this corresponds to an initial delay of 11ms in the ER after the direct sound, and 185ms in the LR.

Advantages Most real-time research systems [Raghuvanshi et al. 2010; Mehra et al. 2013; Taylor et al. 2009] convolve a separate, costly filter with each source signal. Using a few fixed and pre-transformed filters makes our run-time substantially faster. Per-source cost with our scheme is reduced from convolution to six scale-and-sum operations for each CF.

Our method avoids interpolation artifacts with a fast-moving source or listener. When the IR is updated, prior techniques discard as-yet unprocessed output for older filters, which can clip the reverberation. Keeping all past filters active until they exhaust their output is extremely costly. Our approach avoids this problem as each sound source is processed with interpolation weights updated per video frame, effectively rendering it with an up-to-date, untruncated IR.

Our technique integrates easily with current game audio engines, which naturally support feeding a linear combination of signals to a few fixed filters. In game audio terminology, the linear combination is performed by buses that sum their inputs; our CFs are "effects" on the buses and per-source scale factors are the bus "send values".

Wideband extrapolation Though we simulate and extract parameters from frequencies only up to $\nu_{\max} = 500\text{Hz}$ due to practical limitations on precompute time, we apply wideband canonical filters respecting the loudness and decay time parameters. Propagation characteristics are thus extended into unmodeled higher frequencies, producing approximate but plausible results.

Spatialization Our spatialization of direct sound is inaccurate whenever the line of sight from source to listener is occluded. Our stereo canonical filters use a fixed random set of reflection directions as described below. This scheme is plausible and has been used in past work [Raghuvanshi et al. 2010], but it omits important directional effects in many scenes especially outdoors. Directional information can be extracted from wave simulations using vector intensity, then explicitly encoded and rendered using the SIRR technique [Merimaa and Pulkki 2005]. This straightforward extension would drastically increase our system's memory usage; the challenge is compact directional encoding.

Scene	# polys (million)	dimensions (m) L	W	H	# boxes	# probes	sim. cyl. r, h (m)	$\Delta_{\mathbf{x_s}}$ v, h (m)	$\Delta_{\mathbf{x}_\ell}$ (m)	spatial compression ratios L_{DS}	L_{ER}	T_{ER}	T_{LR}	**net**	raw (TB)	encoded (MB)	bake (h)
Citadel	0.4	224	187	53	27	1622	45, 20	3, 1.6	2	4.2	4.8	4.0	7.5	4.8	56	44.1	12
Deck	1.9	135	143	37	2	1263	45, 20	3, 1.6	2	6.9	8.2	6.3	12.6	7.9	44	20.8	13
Sanctuary	2.4	181	151	34	5	1613	45, 20	3, 1.6	2	4.6	5.1	4.2	7.8	5.1	56	41.1	15
Necropolis	2.1	358	169	31	42	2405	45, 15	4, 1.6	2	3.7	4.2	4.1	6.1	4.4	66	53.7	20
Foliage	2.6	144	149	15	3	662	35, 11	2, 1.8	1	6.0	8.0	8.3	11.1	8.0	9	33.0	4

Table 2: Precomputation statistics. For each demo scene, we report the number of polygons, bounding box dimensions for union of all simulation cylinders across probes, the number of region-of-interest boxes, the number of source probes, size of the simulation cylinder (radius, height), source sample spacing (vertical, horizontal), listener sample spacing, spatial compression ratios, estimated raw simulation data size, final encoded size, and total "bake" time in hours.

6 Canonical Filter Generation

Each set of stereo CFs, $\{H_j^{ER}\}$ or $\{H_j^{LR}\}$, must satisfy three properties: it must allow linear interpolation, each of its members j must have appropriately normalized energy, and each must match a specific energy decay profile.

Early reflection CFs, H_j^{ER}, are generated as a sum of specular (S_j) and diffuse (D_j) components such that $S_j + D_j$ has unit energy and matches the targeted exponential energy decay curve, denoted $E_j(t)$. The diffuse signal is initialized to $D_j = t^2\, G(t)$, $t \in [0, \Delta_{ER}]$, where G is Gaussian white noise with zero mean and unit variance. Next, D_j is scaled to constitute 10% of the target energy of 1. Now we divide D_j into non-overlapping time bins, 10ms each. Within each bin, if D_j's energy is larger than $E_j(t)$, we scale its signal to match that smaller energy. The resulting signal's energy quadratically increases in time until it meets $E_j(t)$, after which it exponentially decays to match $E_j(t)$.

The specular signal S_j then accounts for the energy difference between D_j and $E_j(t)$. It is sparse, generated using a set of 250 peaks with random amplitudes and prime number sample delays. This standard technique [Valimaki et al. 2012] minimizes coloration artifacts from periodic delays (Figure 6c). Within each 10ms time bin as above, the signal S_j is scaled so that its energy equals the difference of the bin's target energy determined by $E_j(t)$, and the diffuse energy computed from D_j. So specular peaks in the summed result $S_j + D_j$ decrease in amplitude over time and smoothly disappear into an entirely diffuse signal, as shown in Figure 6. Each ER filter thus makes a smooth transition to the subsequent LR.

To generate stereo filters, each peak in S_j is assigned a random incidence direction, spatialized using the cardioid directivity function, and accumulated into the corresponding (left/right) channel of the output. Diffuse stereo is generated by computing two channels of $G(t)$ with different random number seeds. The left and right channels of the diffuse and specular components are then summed.

To support linear interpolation, all CFs $\{H_j^{ER}\}$ share the same peak delays in the specular filter S_j and the same directions for spatialization. Diffuse noise $G(t)$ is computed independently for each stereo channel but shared across j.

Late reverberation CFs, H_j^{LR}, are generated as Gaussian white noise with an exponential envelope determined by their respective decay rate [Valimaki et al. 2012]. As with the ER, the underlying noise signals are decorrelated between channels but shared across j. We modify the LR filters to model frequency-dependent atmospheric attenuation. A filter sample at time t corresponds to the propagation distance $d = ct$. Using ISO 9613-1, we can compute the per-frequency attenuation at distance d. We use a sliding window to compute STFTs, filter each window via atmospheric absorption at d, and accumulate results over all windows. The absorption model assumes a temperature of 20° C, atmospheric pressure of 101.325Pa and 50% relative humidity. Finally, each channel is normalized to have unit energy in its first 25ms.

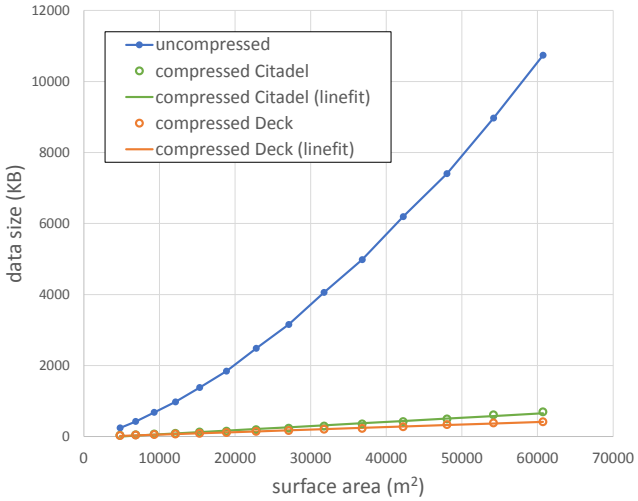


Figure 7: Observed memory scaling with scene size. Our encoding scheme scales linearly in the surface area A of the cylindrical simulation domain, which is proportional to the included surface area of the scene. Uncompressed storage scales with volume, implying super-linear growth via $A^{1.5}$. Results for two scene examples are plotted. For the Citadel scene, the source was placed outside the cathedral as shown in Figure 9, second row. For Deck, it was located as shown in Figure 1.

7 Results

Figures 1 and 9 show parameter fields for three of our scenes. See the supplemental video for real-time captures of our system.

Implementation We precompute on a 140 machine cluster. Computation time is comparable to high-quality light-baking in games (see Table 2); the same compute clusters can be re-used for acoustics. Our run-time is integrated with Unreal Engine 3™. Parameters are decoded (Section 5.1) in the main game thread and converted to send values. Parameters are rendered (Section 5.2) via a custom XAudio2 effect (APO) that implements partitioned convolution using vectorized (SIMD) FFT and complex-vector multiplication. At game-load time, the six canonical filters are read from disk to initialize corresponding effect instances. Effect ownership is transferred to the XAudio2 engine, which runs in a separate thread and executes our filters as part of its audio processing loop.

Memory use Our technique is practical on large, complex scenes with millions of polygons (Table 2). The supplemental video shows acoustic effects in these scenes. Storage for acoustic data is less than 54MB. This is an extremely large compression factor relative to the raw wave field. For Sanctuary, 56TB of raw data encoded to 41MB, a compression factor of over a million. Our encoding automatically allocates less memory to spatially smoother fields; for example, T_{LR} compresses better than L_{DS} in all cases.

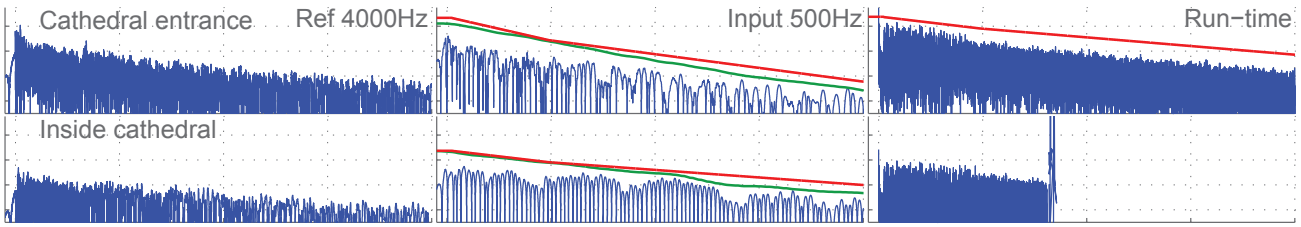


Figure 8: Comparison of simulated vs. run-time IRs. The four rows represent IRs for different listener positions in the Citadel scene, with fixed source located as shown in Figure 9, second row. Left to right, we show the IR from a 4000Hz reference simulation, the 500Hz simulated IR corresponding to the actual input to our system, and the effective run-time IR applied by our system via interpolation of canonical filters. The latter has been low-passed at 4000Hz to aid visual comparison to the reference. The green curve represents the IR's energy decay, using equal weighting over three octave bands (62.5-500Hz). The red curve shows our extracted acoustic parameters, visualized as in Figure 3. We obtain good quantitative agreement between input and rendered acoustic parameters. See the supplementary video for auralization comparisons with the high-frequency reference.

Comparing our memory use to [Raghuvanshi et al. 2010], both techniques sample in the space of source×listener locations with similar spatial resolution. We therefore compare cost per sampled IR. We represent each sample with 4 one-byte parameters. Spatial compression is typically 4-8×, yielding [0.5-1.0] bytes. [Raghuvanshi et al. 2010, page 6] stores 20-50 peaks as pairs of single-precision delay and amplitude, and a residual frequency trend for 10 octave bands. The per-sample cost is [20-50] $\cdot\ 2 \cdot 4 + 10 \cdot 4 =$ [200-440] bytes. The reduction is substantial: 200-880×.

Memory scaling As scenes get larger, we have experimentally verified that the resulting encoded size for each probe scales as the scene's surface area rather than its volume. This indicates that our method has removed one spatial dimension from the problem in addition to time, going from 7D ($volume \times volume \times time$) to 5D ($volume \times area$). Our experiment scaled up the cylindrical simulation region along with the contained scene geometry and recorded the resulting encoded size. (For a linear scale factor l, the number of simulation cells scales as l^3: wave simulations over bigger volumes are costlier to compute.) Figure 7 shows the result for two scenes, Citadel and Deck. In both cases, the encoded size scales linearly with the surface area of the bounding cylinder. The unencoded size scales with the scene volume, and so has super-linear growth in surface area. The Kirchoff-Helmholtz integral theorem for monochromatic wave fields suggests that scaling by surface area may be optimal in general scenes, since it is proportional to the information content of the boundary conditions.

Performance Parameter decoding (including decompression and visibility-aware 6D interpolation) is very fast. We benchmarked the parameter decode step based on 100,000 random pairs-of-points and observed an average compute time of about 100μs on an Intel Xeon E5540@2.53GHz, for all scenes. This is a pessimistic estimate which obviates our cache. The calculation can be performed on the same main core in which the game runs, supporting ~100 active sounds due to numerous agents making footsteps, gunshots and other sounds, along with ambient sounds. The acoustic parameters are updated every frame, contributing to the smoothness of the sound even with fast moving sources, as illustrated in the Necropolis segment in the supplementary video.

Our decoding performance is similar to that in [Raghuvanshi et al. 2010]; both systems perform spatial lookup and interpolation. Our convolutional rendering, which comprises most of the run-time for both techniques, is faster as discussed in Section 5.2. On-the-fly geometric acoustics systems supporting dynamic geometry, such as [Taylor et al. 2009](Table 1) and [Schröder 2011](Table 6.1, Fig. 11.7), require a second or more of simulation time using multiple cores to compute the IR of a single moving source, on simple models (400 and 50,000 polygons respectively) with restricted path order (~3). We robustly handle practical game scenes containing millions of polygons. Our method, while restricted to static scenes, is 10,000× faster, within the audio budget for compute-intensive desktop graphical applications.

Comparison to an uncompressed reference We compare our system's run-time results to the unencoded ($\nu_{\max} = 500$Hz) simulation we precomputed, as well as to a higher-frequency ($\nu_{\max} = 4000$Hz) reference simulation. Atmospheric attenuation, neglected by our simulator, is applied using the technique in Section 6. The results of the 4000Hz simulation were scaled to match the 500Hz simulation's octave-averaged loudness in the 0-500Hz range. In Figure 8, we compare parameters re-extracted from the rendered run-time IR with that from the raw 500Hz simulation, as well as an octave-averaged energy decay curve. Since simulation does not directly provide directional information, we compare against monaural canonical filters. We use the unspatialized sum $S_j + D_j$ (Section 6) for the ER and choose a single stereo channel for the LR.

Across our entire system including encoding, spatial interpolation of parameters, and application of interpolated canonical filters, we observe mean errors of 3.3dB and 22.5% for the loudness and decay rate parameters respectively, over all points shown in the supplementary video. These errors are close to our chosen quantization levels, also compared visually as the red traces in Figure 8. Refer to the supplementary video to hear the comparison. While our system reproduces spatial variation in occlusion and reverberation effects well, it sounds overly "bright" because we neglect low-pass filtering effects which accompany occlusion. Furthermore, the reference reverberation exhibits significantly higher decay rates at frequencies above 500Hz, especially when the listener is inside the cathedral.

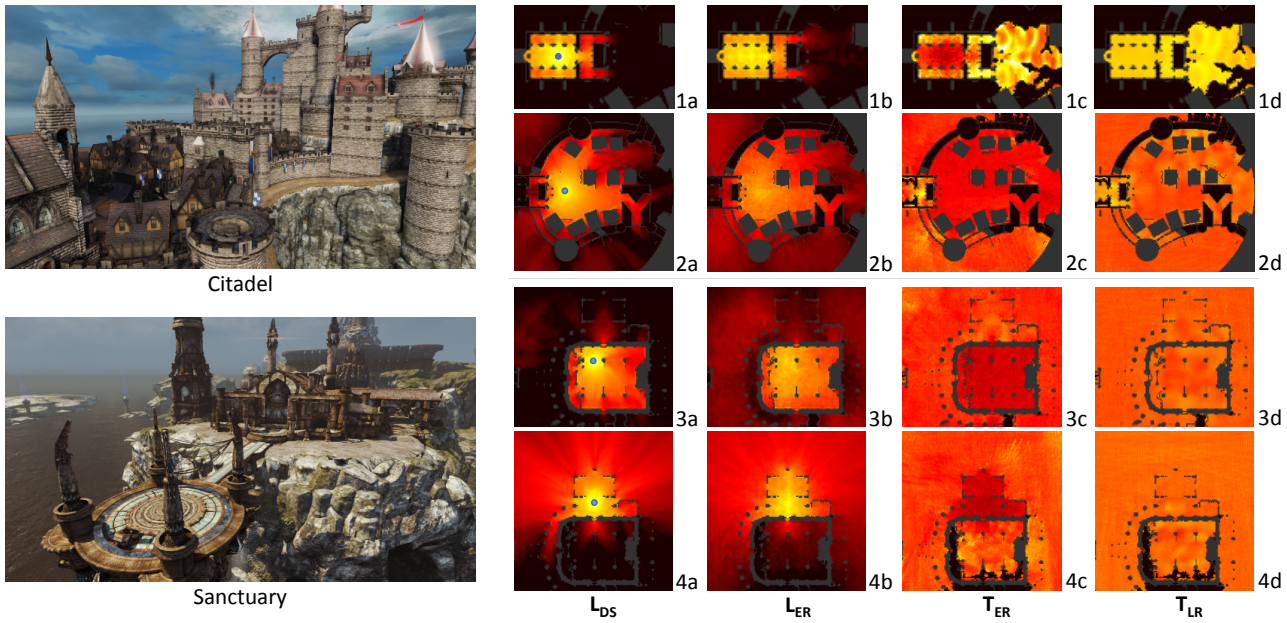


Figure 9: Parameter fields for the Citadel and Sanctuary scenes. A 2D slice of the parameter field is visualized at two fixed probe source points (blue dot) for each scene, indoors (odd rows) and outdoors (even rows). Notice in 1a the shadowing in direct sound (L_{DS}) caused by pillars in the cathedral and the steep falloff by the time sound gets to the exit – each diffraction around an edge substantially reduces its energy. The corresponding L_{ER} field in 1b subsumes a large number of reflected paths and so is smoother inside the cathedral. Notice multiple shadowing in 2a and 4a. Unlike light rays which extinguish after a single hit, a sound wavefront bends around objects and can be shadowed repeatedly. In 4a notice the oval peak around the source: this represents a loudness boost due to ground reflections from an oval platform. Also note the clear far-field shadowing patterns. In all cases, T_{ER} exhibits more spatial variation than T_{LR}. Observe in the top row that T_{ER} and T_{LR} vary significantly outside the cathedral due to slope estimation on weak signals. The low loudness renders this variance inaudible, as shown in the supplementary video.

8 Conclusions and Future Work

We presented a new parametric approach for precomputed sound propagation. By encoding chaotic time-evolving pressure as perceptual parameter fields, it greatly reduces computation and memory demands and becomes practical for many applications including 3D games. Our approach has two advantages. First, it removes the wave field's time dependence and exposes its perceptual smoothness, exploited by our compressor. Second, it allows rendering of acoustic parameters by feeding linear combinations of source signals to a small set of canonical filters. This avoids costly per-source convolution and IR interpolation artifacts, yielding a fast, artifact-free run-time supporting hundreds of sounds.

Much remains for future work. Our current parameter set is based solely on energy decay and could be augmented. Encoding directional information (spatialization) would do much to increase our system's realism. The same is true for the IR's "diffuseness" which we currently fix in our system at 10%, and for the spectral effects we neglect such as low-pass filtering from occlusion and frequency-dependent reverberation. We note that acoustic parameters can be separately coded over multiple octave bands for increased quality at the cost of more memory. Our parameter set is plainly insufficient for outdoor reverberation and echoes; compared to the extensive literature on indoor room acoustics, little work has investigated a perceptually sufficient parameterization for outdoor IRs. We expect our system may be extended to handle limited dynamic geometry such as open/closed doors. Large scene changes such as destroying walls or buildings are precluded by our precomputed approach. Finally, parametric wave field coding might have application to compressing acoustic radiation fields [James et al. 2006].

Acknowledgments

Thanks to Jimmy Smith, John Tennant, John Morgan, and Mike Rayner for their indispensable feedback and guidance. We are grateful to our steadfast collaborator, Kristofor Mellroth, for his advice, help, and expert ears.

References

AJDLER, T., SBAIZ, L., AND VETTERLI, M. 2006. The Plenacoustic Function and Its Sampling. *Signal Processing, IEEE Transactions on 54*, 10 (Oct.), 3790–3804.

ALLEN, J. B., AND BERKLEY, D. A. 1979. Image method for efficiently simulating small-room acoustics. *J. Acoust. Soc. Am 65*, 4, 943–950.

BERANEK, L. L. 2003. Subjective Rank-Orderings and Acoustical Measurements for Fifty-Eight Concert Halls. *Acta Acustica united with Acustica* (May), 494–508.

BRADLEY, J. S., AND HALLIWELL, R. E. 1988. Accuracy and reproducibility of auditorium acoustics measures. In *British Institute of Acoustics*, vol. 10, 399–406.

CALAMIA, P. 2009. *Advances in Edge-Diffraction Modeling for Virtual-Acoustic Simulations*. PhD thesis, Princeton University.

CHANDAK, A., LAUTERBACH, C., TAYLOR, M., REN, Z., AND MANOCHA, D. 2008. AD-Frustum: Adaptive Frustum Tracing for Interactive Sound Propagation. *IEEE Transactions on Visualization and Computer Graphics 14*, 6, 1707–1722.

DAVY, J. L., DUNN, I. P., AND DUBOUT, P. 1979. The Variance of Decay Rates in Reverberation Rooms. *Acta Acustica united with Acustica* (Aug.), 12–25.

FUNKHOUSER, T., TSINGOS, N., CARLBOM, I., ELKO, G.,

SONDHI, M., WEST, J. E., PINGALI, G., MIN, P., AND NGAN, A. 2004. A beam tracing method for interactive architectural acoustics. *The Journal of the Acoustical Society of America 115*, 2, 739–756.

GADE, A. 2007. Acoustics in Halls for Speech and Music. In *Springer Handbook of Acoustics*, T. Rossing, Ed., 2007 ed. Springer, May, ch. 9.

IASIG, 3D WORKING GROUP. 1999. *Interactive 3D Audio Rendering Guidelines, Level 2.0*, Sept.

ISO 3382-1:2009. Acoustics - Measurement of room acoustic parameters - Part 1: Performance spaces. *International Organization for Standardization*.

JAMES, D. L., BARBIC, J., AND PAI, D. K. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics 25*, 3 (July), 987–995.

KOLARIK, A. J., CIRSTEA, S., AND PARDHAN, S. 2011. Perceiving auditory distance using level and direct-to-reverberant ratio cues. *The Journal of the Acoustical Society of America 130*, 4 (Oct.), 2545.

KOWALCZYK, K., AND VAN WALSTIJN, M. 2010. Room acoustics simulation using 3-D compact explicit FDTD schemes. *IEEE Transactions on Audio, Speech and Language Processing*.

KROKSTAD, A. 2008. The Hundred Years Cycle in Room Acoustic Research and Design. In *Reflections on sound*, P. Svensson, Ed. Norwegian University of Science and Technology (NTNU), Trondheim, Norway, June, ch. 5.

KUTTRUFF, H. 2000. *Room Acoustics*, 4 ed. Taylor & Francis.

LAINE, S., SILTANEN, S., LOKKI, T., AND SAVIOJA, L. 2009. Accelerated beam tracing algorithm. *Applied Acoustics 70*, 1 (Jan.), 172–181.

MEHRA, R., RAGHUVANSHI, N., SAVIOJA, L., LIN, M. C., AND MANOCHA, D. 2012. An efficient GPU-based time domain solver for the acoustic wave equation. *Applied Acoustics 73*, 2 (Feb.), 83–94.

MEHRA, R., RAGHUVANSHI, N., ANTANI, L., CHANDAK, A., CURTIS, S., AND MANOCHA, D. 2013. Wave-based Sound Propagation in Large Open Scenes Using an Equivalent Source Formulation. *ACM Trans. Graph. 32*, 2 (Apr.).

MERIMAA, J., AND PULKKI, V. 2005. Spatial Impulse Response Rendering I: Analysis and Synthesis. *J. Audio Eng. Soc 53*, 12, 1115–1127.

MURPHY, D., KELLONIEMI, A., MULLEN, J., AND SHELLEY, S. 2007. Acoustic Modeling Using the Digital Waveguide Mesh. *IEEE Signal Processing Magazine 24*, 2 (Mar.), 55–66.

RAGHUVANSHI, N., NARAIN, R., AND LIN, M. C. 2009. Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition. *IEEE Transactions on Visualization and Computer Graphics 15*, 5, 789–801.

RAGHUVANSHI, N., SNYDER, J., MEHRA, R., LIN, M. C., AND GOVINDARAJU, N. K. 2010. Precomputed Wave Simulation for Real-Time Sound Propagation of Dynamic Sources in Complex Scenes. *ACM Transactions on Graphics (proceedings of SIGGRAPH 2010) 29*, 3 (July).

RINDEL, J. H., AND CHRISTENSEN, C. L. 2013. The use of colors, animations and auralizations in room acoustics. In *Internoise 2013*.

SABINE, H. 1953. Room acoustics. *Audio, Transactions of the IRE Professional Group on 1*, 4, 4–12.

SAKAMOTO, S., NAGATOMO, H., USHIYAMA, A., AND TACHIBANA, H. 2008. Calculation of impulse responses and acoustic parameters in a hall by the finite-difference time-domain method. *Acoustical Science and Technology 29*, 4.

SAVIOJA, L., RINNE, T., AND TAKALA, T. 1994. Simulation of room acoustics with a 3-D finite difference mesh. In *Proceedings of the International Computer Music Conference*, 463–466.

SAVIOJA, L. 2010. Real-Time 3D Finite-Difference Time-Domain Simulation of Mid-Frequency Room Acoustics. In *13th International Conference on Digital Audio Effects (DAFx-10)*.

SCHRÖDER, D. 2011. *Physically Based Real-Time Auralization of Interactive Virtual Environments*. Logos Verlag, Dec.

SILTANEN, S., LOKKI, T., KIMINKI, S., AND SAVIOJA, L. 2007. The room acoustic rendering equation. *The Journal of the Acoustical Society of America 122*, 3 (Sept.), 1624–1635.

SILTANEN, S., LOKKI, T., AND SAVIOJA, L. 2009. Frequency Domain Acoustic Radiance Transfer for Real-Time Auralization. *Acta Acustica united with Acustica 95*, 1, 106–117.

SILTANEN, S., LOKKI, T., AND SAVIOJA, L. 2010. Rays or Waves? Understanding the Strengths and Weaknesses of Computational Room Acoustics Modeling Techniques. In *Proc. Int. Symposium on Room Acoustics*.

SILTANEN, S., LOKKI, T., AND SAVIOJA, L. 2010. Room acoustics modeling with acoustic radiance transfer. *Proc. ISRA Melbourne*.

SILTANEN, S. 2005. *Geometry Reduction in Room Acoustics Modeling*. Master's thesis, Helsinki University of Technology.

SOUTHERN, A., SILTANEN, S., MURPHY, D. T., AND SAVIOJA, L. 2013. Room Impulse Response Synthesis and Validation Using a Hybrid Acoustic Model. *Audio, Speech, and Language Processing, IEEE Transactions on 21*, 9, 1940–1952.

STEPHENSON, U. M., AND SVENSSON, U. P. 2007. An improved energetic approach to diffraction based on the uncertainty principle. In *19th Int. Cong. on Acoustics (ICA)*.

STETTNER, A., AND GREENBERG, D. P. 1989. Computer Graphics Visualization for Acoustic Simulation. *SIGGRAPH Comput. Graph. 23*, 3 (July), 195–206.

SVENSSON, U. P., FRED, R. I., AND VANDERKOOY, J. 1999. An analytic secondary source model of edge diffraction impulse responses. *The Journal of the Acoustical Society of America 106*, 5 (Nov.), 2331–2344.

TAKALA, T., AND HAHN, J. 1992. Sound rendering. *SIGGRAPH Comput. Graph. 26*, 2 (July), 211–220.

TAYLOR, M. T., CHANDAK, A., ANTANI, L., AND MANOCHA, D. 2009. RESound: interactive sound rendering for dynamic virtual environments. In *Proceedings of ACM conference on Multimedia*, ACM, New York, NY, USA, 271–280.

TSINGOS, N. 2009. Pre-computing geometry-based reverberation effects for games. In *35th AES Conference on Audio for Games*.

VALIMAKI, V., PARKER, J. D., SAVIOJA, L., SMITH, J. O., AND ABEL, J. S. 2012. Fifty Years of Artificial Reverberation. *Audio, Speech, and Language Processing, IEEE Transactions on 20*, 5 (July), 1421–1448.

YEH, H., MEHRA, R., REN, Z., ANTANI, L., MANOCHA, D., AND LIN, M. 2013. Wave-ray Coupling for Interactive Sound Propagation in Large Complex Scenes. *ACM Trans. Graph. 32*, 6 (Nov.).