High-Order Diffraction and Diffuse Reflections for Interactive Sound Propagation in Large Environments

Carl Schissler[*] Ravish Mehra[†] Dinesh Manocha[‡]
University of North Carolina at Chapel Hill

Figure 1: *Our high-order diffraction and diffuse reflection algorithms are used to generate plausible sound effects at interactive rates on large static and dynamic scenes: (left) interior office (154K triangles); (center) oil refinery (245K triangles); (right) city (254K triangles).*

Abstract

We present novel algorithms for modeling interactive diffuse reflections and higher-order diffraction in large-scale virtual environments. Our formulation is based on ray-based sound propagation and is directly applicable to complex geometric datasets. We use an incremental approach that combines radiosity and path tracing techniques to iteratively compute diffuse reflections. We also present algorithms for wavelength-dependent simplification and visibility graph computation to accelerate higher-order diffraction at runtime. The overall system can generate plausible sound effects at interactive rates in large, dynamic scenes that have multiple sound sources. We highlight the performance in complex indoor and outdoor environments and observe an order of magnitude performance improvement over previous methods.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.5 [Computer Graphics]: Applications—Sound rendering;

Keywords: sound propagation, diffuse reflections, diffraction, simplification

Links: DL PDF WEB VIDEO

1 Introduction

Virtual environment technologies are widely used in different applications, including engineering design, training, architecture, and entertainment. In order to improve realism and immersion, it is important to augment visual perceptions with matching sound stimuli and auralize the sound fields. The resulting auditory information can significantly help the user evaluate the environment in terms of spaciousness and sound localization.

In this paper, we address the problem of interactive sound propagation and rendering in large-scale virtual environments composed of multiple moving sources and objects. These include large urban environments spanning kilometers and made up of tens or hundreds of buildings with multiple moving vehicles. Other scenarios include large indoor environments such as auditoriums, offices, or factories with volumes up to tens or hundreds of thousands of cubic meters. The model complexity and large dimensions of these spaces result in many acoustic effects including reflections, scattering between the objects, high-order diffraction, late reverberation, echoes, etc.

The most accurate propagation algorithms for modeling various acoustic effects are based on numerically solving the acoustic wave equation. However, the complexity of these methods increases as a linear function of the surface area of the primitives or the volume of the acoustic space, and as at least a cubic function of the maximum simulated frequency. Recently, many wave-based precomputation techniques have been proposed for interactive applications [James et al. 2006; Tsingos et al. 2007; Raghuvanshi et al. 2010; Mehra et al. 2013; Yeh et al. 2013]. However, current algorithms are limited to static scenes and the computational and memory requirements increase significantly for large virtual environments.

Some of the widely used techniques for interactive sound propagation are based on geometric acoustics (GA) and use computations based on ray theory. These are used to compute early reflections and diffractions in static scenes [Funkhouser et al. 1998; Tsingos et al. 2001; Chandak et al. 2009] or to precompute reverberation effects [Tsingos 2009; Antani et al. 2012b]. A major challenge is to extend these GA techniques to complex virtual worlds with multiple moving objects or sources. In a large environment, surface scattering and edge diffraction components tend to overshadow specular reflections after a few orders of reflection [Kuttruff 1995]. Recent advances in ray tracing are used to develop fast sound propagation algorithms for dynamic scenes [Lentz et al. 2007; Pelzer and Vorländer 2010; Taylor et al. 2012], but these methods still cannot compute compute high-order edge diffraction or diffuse reflections at interactive rates.

[*]schissle@cs.unc.edu
[†]ravish.mehra07@gmail.com
[‡]dm@cs.unc.edu

Main Results: We present a novel geometric ray-based interactive sound propagation approach to generating plausible sound effects in large, complex virtual environments. Our formulation is directly applicable to widely-available, detailed geometric datasets. We compute early reflections and diffractions using GA and late reverberation using statistical techniques to automatically handle large dynamic scenes. In order to achieve interactive performance, we present three new algorithms (Section 3):

1. **Interactive diffuse reflections:** We present an iterative approach that uses a combination of path tracing and radiosity techniques to compute diffuse reflections. We exploit spatial and temporal coherence to reuse some of the rays traced during previous frames. We observe an order of magnitude improvement over prior algorithms.
2. **Higher-Order Edge Diffraction:** We precompute a global edge visibility graph. At runtime, we traverse the graph and compute the higher-order diffraction contributions based on the uniform theory of diffraction.
3. **Automatic Simplification for Edge Diffraction:** We present a wavelength-dependent simplification scheme to significantly reduce the number of diffraction edges in a complex scene. This results in a first practical algorithm to perform higher-order diffraction in complex scenes.

We demonstrate the performance of our algorithm in large urban scenes with tens of buildings, as well as complex indoor scenes corresponding to factories and offices with hundreds of obstacles (Section 4). The performance scales with the number of cores, and we are able to perform interactive sound propagation and rendering at $15 - 50$ frames per second using a 4-core CPU. Our approach scales logarithmically with the model complexity of the scene and linearly with the number of moving sources and objects. Its accuracy for indoor scenes is comparable to prior GA techniques (Section 5). To the best of our knowledge, this is the first approach that can generate plausible acoustic effects for large and complex virtual environments at interactive rates.

2 Related Work

In this section, we give a brief overview of prior work on sound propagation. There is considerable literature on acoustical modeling and propagation techniques for room acoustics [Kuttruff 2007] and outdoor environments [Attenborough et al. 2007]. The most accurate algorithms for sound propagation are based on solving the acoustic wave equation using numerical techniques, including finite-difference time-domain method [Savioja 2010], the finite-element method [Thompson 2006], the boundary-element method [Gumerov and Duraiswami 2009], adaptive rectangular decomposition [Raghuvanshi et al. 2010], the equivalent source method [Mehra et al. 2013], etc. These are primarily used for lower frequencies, and combined with GA techniques to simulate for higher frequencies [Yeh et al. 2013]. Other recent work enables source and listener directivity for wave-based acoustics [Mehra et al. 2014]. These techniques are limited to static scenes, and the precomputation cost can be extremely high (orders of petaflops) for large virtual environments.

GA techniques have been around since 1950s and have been extensively studied. The commonly used techniques are based on image source methods [Allen and Berkley 1979; Borish 1984] and ray tracing [Krokstad et al. 1968; Vorländer 1989]. Many algorithms have also been designed for interactive sound propagation based on beam tracing [Funkhouser et al. 1998], frustum tracing [Taylor et al. 2009; Chandak et al. 2009], and ray tracing [Lentz et al. 2007; Taylor et al. 2012; Schissler and Manocha 2011]. Aural proxy objects have also been investigated for efficiently computing early reflections, and local visibility and depth used to approximate directional late reverberation [Antani and Manocha 2013].

Some of the commonly used techniques to approximate edge diffraction are based on the uniform theory of diffraction (UTD) [Kouyoumjian and Pathak 1974] and the Biot-Tolstoy-Medwin (BTM) model [Svensson et al. 1999]. However, current interactive diffraction algorithms are limited to static scenes [Tsingos et al. 2001; Antani et al. 2012a] or first-order diffraction in dynamic scenes [Taylor et al. 2012]. Many techniques have been proposed for diffuse acoustic reflections based on ray tracing [Lentz et al. 2007; Alarcao et al. 2009; Taylor et al. 2009] and the acoustic rendering equation [Siltanen et al. 2007; Antani et al. 2012b]. Other interactive algorithms use multi-resolution [Wand and Straßer 2004] and perceptual [Moeck et al. 2007] techniques to accelerate audio rendering of complex scenes.

3 Overview

In this section, we present novel techniques to compute fast diffuse reflections, higher-order edge diffraction, and automatic simplification of large datasets. Ray tracing has been widely used for offline and interactive sound propagation [Krokstad et al. 1968; Vorländer 1989; Bertram et al. 2005; Taylor et al. 2012]. In ray tracing, propagation paths are computed by generating rays from each source or receiver position and propagating them through the scene, modeling reflection and diffraction effects.

Our approach is targeted towards large and spacious models, and assumes homogeneous media and a constant sound speed. We use geometric acoustic (GA) techniques to accurately compute early reflections (e.g. up to 10 orders) and assume that the surface primitives are large compared to the wavelength. We use statistical methods to compute late reverberation.

We build on recent advances in interactive ray tracing for visual and sound rendering. We use ray tracing to accelerate the image-source method for computing early specular reflections [Vorländer 1989] and use the uniform theory of diffraction (UTD) to approximate edge diffraction. Frequency-dependent effects are modeled using different absorption and scattering coefficients for discrete frequency bands.

3.1 Iterative Diffuse Reflections

A diffuse reflection occurs when sound energy is scattered into non-specular directions. The diffuse sound-energy density w at any point $\vec{p}$ in space at a time t in is given by Equation (1), where L' is the distance from the surface element dS' to the listener, ς'' is the angle of the sound wave which radiates from the surface element dS', $\alpha(\vec{p'})$ is the reflection coefficient as a function of $\vec{p}$, B is the irradiation strength, c is the speed of sound, and $w_d(\vec{p}, t)$ is the direct sound contribution from the sound source [Embrechts 2000; Nironen 2004]:

$$w(\vec{p}, t) = \frac{1}{\pi c} \int \int_S \alpha(\vec{p'}) B \left(\vec{p'}, t - \frac{L'}{c} \right) \frac{\cos \varsigma''}{L'^2} dS' + w_d(\vec{p}, t). \tag{1}$$

In order to handle frequency-dependent absorption, $\alpha(\vec{p'})$ may be represented as a vector of attenuation values for discrete frequency bands. In sound rendering it is important to model the time and phase dependence of sound waves. The time dependence is represented by the L'/c term that computes the delay time due to that propagation along that path. This delay time is used to determine the phase relationship between the original and reflected sound and is responsible for producing acoustic phenomena like echoes.

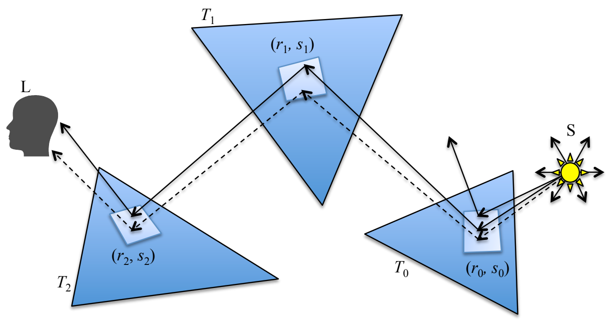

Figure 2: *An example set of 3rd-order diffuse ray paths. Rays leave the sound source S, hit the sequence of surface patches* $\{T_0(r_0, s_0), T_1(r_1, s_1), T_2(r_2, s_2)\}$*, then hit the listener L. Rays with dashed paths are from previous frames, while rays with solid paths are from the current frame. Our diffuse reflection algorithm groups these coherent rays together because they hit the same sequence of surface patches. The sound contribution at the listener is averaged over the time period, using rays from the previous and current frames.*

Since there is no closed-form solution for Equation (1) for general scenes, traditional diffuse sound algorithms approximate this integral using numerical techniques. A commonly used method, diffuse path tracing [Embrechts 2000], traces many random rays from each sound source and diffusely reflects these rays through the scene to solve the acoustic rendering equation [Siltanen et al. 2007; Antani et al. 2012b]. An intersection test is performed for each ray to calculate its intersection with the listener, who is represented by a sphere the size of a human head. Rays that intersect with a given listener position contribute to the final impulse response for that sound source at that listener's location. The path tracing algorithm can generate accurate results and is frequently used for offline acoustic simulation. Since diffuse path tracing is a Monte-Carlo method, it requires a very high number of ray samples to generate accurate results. Therefore, current techniques for interactive diffuse reflections are limited to very simple models and can only compute $1-3$ orders of reflections [Lentz et al. 2007; Taylor et al. 2009]. Some extensions have been proposed such as "diffuse rain" [Dross et al. 2007], which can drastically increase the number of ray contributions by generating an additional reflection from each ray hit point to the listener.

In order to accelerate diffuse reflection computation, we use ideas from radiosity algorithms that are widely used in visual and sound rendering. Radiosity is an alternate method to path tracing that models diffuse reflections by decomposing the scene into small surface patches, computing view factors (or form factors) for each pair of patches, and computing the intensity for each patch as the sum of the contributions from all other patches. Radiosity has also been used to compute sound fields [Franzoni et al. 2001]. These approaches discretize the inner integral of Equation (1) into the following equation for a single surface element [Nironen 2004]:

$$I_i(t) = \sum_{j:j\neq i} m_{j\rightarrow i}\alpha_j I_j\left(t - \frac{p_{j\rightarrow i}}{c}\right)\Delta S_j + I_{0\rightarrow i}(t), \qquad (2)$$

where $I_i(t)$ is the incident sound intensity at surface patch i at time t, $I_{0\rightarrow i}(t)$ is the direct contribution from the source at patch i at time t, I_j is the contribution from a surface patch j, and $m_{j\rightarrow i}$ is the view factor between patches j and i. The surface intensities for all patches in the scene are added to compute the resulting sound field at a listener location $\vec{p}$ at time t:

$$w(\vec{p}, t) = \sum_i I_i(t)v_i(\vec{p}, t), \qquad (3)$$

where $v_i(\vec{p}, t)$ is the visibility function for patch i that has range $[0, 1]$, which indicates the fraction of that patch visible to point $\vec{p}$. This formulation of sound-field computation benefits from less sampling noise than path tracing, but it also requires a high degree of surface subdivision to accurately solve the acoustic rendering equation. In addition, current radiosity-based algorithms are mainly limited to static environments, because recomputing view factors at runtime is expensive, and because the memory and time complexity grows with the surface area of the scene. This makes them unsuitable for large-scale interactive diffuse sound propagation in dynamic scenes.

Our *approach* combines path tracing with radiosity-like patch subdivision to reduce sampling noise for interactive diffuse reflections. We reuse the rays traced during previous frames for the current frame. We assume that the changes in the locations of sound sources, listeners, and dynamic obstacles are small between successive frames. Therefore, rays that hit the same sequence of surface patches during different frames are grouped together. The grouped rays' contributions are summed to compute a better estimate of the reflected sound intensity I_i for that sequence of surface patches, as shown in Fig. 2. Compared with standard path tracing, our approach reduces the number of rays required to compute accurate diffuse sound and improves temporal coherence of the resulting sound field. By shooting fewer rays per frame, we reduce computation time and improve the interactivity of path tracing for sound propagation.

Visual vs. Sound Rendering: The use of frame-to-frame coherence along with combining path tracing and radiosity methods has been investigated in visual rendering [Krivanek et al. 2008]. This includes caching of diffuse illuminance data (e.g. irradiance caching) to accelerate the computation of global illumination. The notion of reusing ray paths has been used in visual rendering techniques based on frameless rendering and progressive refinement. However, sound rendering differs from visual rendering in several ways. In particular, sound rendering involves computation of phase and time delay information, which results in different formulations. Additionally, radiosity algorithms for visual rendering require a fine subdivision to capture abrupt changes in the view factor such as with hard shadows [Krivanek et al. 2008]. On the other hand, the incoherence of diffuse sound rays implies that changes in the incident intensity are usually gradual. This allows us to use larger surface patches in sound rendering [Nironen 2004].

3.1.1 Subdivision

As part of a preprocessing step, we subdivide the triangles in the scene into a set of surface patches. This operation can also be done efficiently at runtime if the scene geometry deforms. For the subdivision, we ensure that each patch is roughly the same size and meets minimum spatial size criteria. We use Barycentric coordinates to partition each triangle in the input scene into a grid of quadrilateral and triangular patches. Patches are arranged as a 2-dimensional grid of entries with indices (r, s), as shown in Fig. 3. We do not store these patches explicitly; instead we use the Barycentric coordinates of each ray-triangle intersection, along with precomputed information about the triangle, to determine which surface patch contains the intersection point at runtime. This formulation requires only a few extra bytes per triangle.

In order to precompute the subdivision for a given triangle T, we select a vertex $\mathbf{k}$ of T as the *key* vertex for that triangle: the vertex is the one that is incident to the longest altitude of T. The length of the altitude from $\mathbf{k}$, d, is used to determine the number of rows in the subdivision $n_r = \lceil d/l \rceil$, where l is a parameter used to govern the resolution of the subdivision. In addition, the number of columns for the largest row is $n_s = \lceil e/l \rceil$ where e is the length of

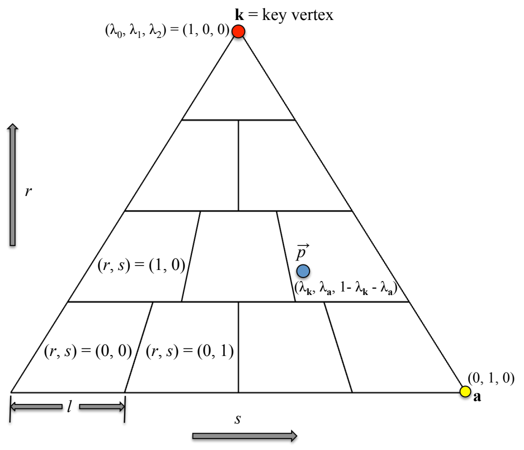

Figure 3: *An example triangle subdivision. The triangle is subdivided into an array of indexed patches (r, s) based on the subdivision resolution l. We compute the ray intersection point $\vec{p}$ with Barycentric coordinates $(\lambda_k, \lambda_a, 1 - \lambda_k - \lambda_a)$, (e.g. $(r, s) = (1, 3)$).*

the edge opposite $\mathbf{k}$. The number of columns n_s^r for the rth row is determined by

$$n_s^r = \left\lceil \frac{n_r - r}{n_r} n_s \right\rceil . \quad (4)$$

In order to determine this subdivision at runtime, we only store the values of n_r, n_s, and the index of key vertex $\mathbf{k}$ for each triangle. The choice of subdivision size l determines the size of the patches and accuracy of the approach, as in radiosity-based algorithms. In general, l should be chosen such that the incident sound intensity doesn't change too much across adjacent patches.

3.1.2 Diffuse Path Cache

We maintain a separate hash-table cache of diffuse reflectance data for each sound source. This cache is used to store the combined contributions of many sound rays from previous frames that are grouped based on the surface subdivision. Each cache entry corresponds to a unique series of surface patches $\{T_0(r_0, s_0), ..., T_n(r_n, s_n)\}$, where each element of the series indicates one of the triangles T_i and a surface patch (r_i, s_i) on T_i. This entry represents the $n + 1$ diffuse reflections that have occurred for rays emitted along the path to the listener.

Each cache entry also stores the set of values $\{\eta, \mu, \hat{\alpha}, \hat{\delta}\}$, η is the number of rays following this entry's path that have hit the listener; μ is the total number of rays emitted from the source for all frames, while this entry was in the cache; $\hat{\alpha} = \sum \alpha$ is the sum of the total frequency-dependent attenuation $\alpha \in [\mathbf{0}, \mathbf{1}]$ (due to the $n + 1$ diffuse reflections for all rays that have traveled the path for this entry); and $\hat{\delta} = \sum \delta$ is the sum of the path lengths δ for all rays that have hit this sequence of surface patches while the entry was in the cache. From these values, the average incident sound source intensity I_i for this patch sequence i received at the listener as a fraction of the total emitted energy can be computed by:

$$I_i = \frac{\eta}{\mu} \frac{\hat{\alpha}}{\eta}. \quad (5)$$

The value of η/μ estimates the average of the combined $m_{j \to i}$, I_j, and $I_{0 \to i}(t)$ terms from (2). Those terms together determine the frequency-independent fraction of source energy reflected from a surface patch, which is the same value estimated by η/μ. $\hat{\alpha}/\eta$ approximates the average α_j term from (2). To compute the average path length $\bar{\delta}$ for a cache entry, we use:

$$\bar{\delta} = \frac{\hat{\delta}}{\eta}. \quad (6)$$

This average path length is divided by the speed of sound c in the propagation medium to determine the average delay time for this path.

3.1.3 Runtime

At the beginning of each simulation step, we trace random rays from each sound source position and diffusely reflect those rays through the scene to an arbitrary maximum depth (e.g. 10), as in traditional path tracing. For each ray-triangle intersection, we first find the surface patch, $T(r, s)$, for the intersection point $\vec{p}$ on triangle T. We compute the Barycentric coordinates $(\lambda_0, \lambda_1, \lambda_2)$ of $\vec{p}$ with respect to triangle T. Next, we choose two of the three components of the Barycentric coordinates (λ_k, λ_a) from the set $(\lambda_0, \lambda_1, \lambda_2)$ in order to define the subdivision axes. λ_k is the component corresponding to the key vertex $\mathbf{k}$, and λ_a is the component for the vertex $\mathbf{a}$ that is to the left of $\mathbf{k}$ on triangle T. Given λ_k and λ_a, we can compute the row and column indices (r, s) for the surface patch containing $\vec{p}$, as shown in Fig. 3: $r = \lfloor \lambda_k \cdot n_r \rfloor$, $s = \lfloor \lambda_a \cdot n_s^r \rfloor$. This patch $T(r, s)$ is added to the patch sequence for the ray that is currently being propagated.

When the ray is reflected, the outgoing ray is tested to see if it intersects the listener's detection sphere. If so, the sequence of previous surface patches, $\{T_0(r_0, s_0), ..., T_n(r_n, s_n)\}$, where reflections occurred along this path is used to access the diffuse cache. If there was an existing cache entry for that specific patch sequence, the entry is updated with the contribution for that ray:

$$\eta_{new} = \eta + 1; \quad \hat{\alpha}_{new} = \hat{\alpha} + \alpha_{new}; \quad \hat{\delta}_{new} = \hat{\delta} + \delta_{new}. \quad (7)$$

If there is no entry corresponding to this sequence of patches, a new entry is inserted into the cache and the corresponding parameters are set as $\eta = 1$, $\mu = 0$, $\hat{\alpha} = \alpha_{new}$, $\hat{\delta} = \delta_{new}$.

Impulse Response Computation: After all the rays have been traced from the source and the cache entries updated for rays that hit the listener, the cache contains entries that correspond to the accumulated contribution of groups of rays that have traveled along similar paths to the listener during the current frame or previous frames. Next, we compute the final impulse response for this source-listener combination from the cache by iterating through all entries and generating a delayed impulse for each entry. For each entry, the value of μ is increased by the total number of rays emitted from the source during this frame. We use equation (5) to compute the incident intensity I_i for this cache entry. If this intensity value is less than some threshold κ, then very few rays have hit the sequence of surface patches corresponding to the cache entry in recent frames. In this case, the cache entry is removed because it does not significantly contribute to the final impulse response at the listener's location. We use a cutoff threshold of $\kappa = -60$dB or $1/1000$th of the original source's energy; this threshold is commonly used in measuring the reverb time, RT60, of an acoustical space [Eyring 1930]. Cache entries that exceed κ in energy contribute to the output impulse response. The delay time for this entry's contribution is computed using the average path length from equation (6) and the speed of sound. Finally, this contribution is added to the output sound field at the listener's location using equation (3), where the value of the visibility function v_i is always 1 as all of the sound source contributions for the path are known to intersect the listener.

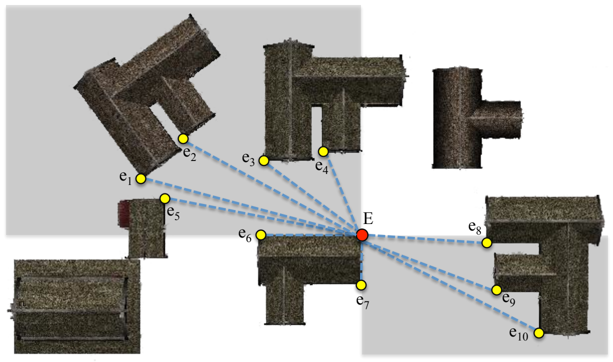

Figure 4: *A top-down view of a portion of a diffraction edge visibility graph for a small village scene, shown here for edge E. Edges $e_{1..10}$ are visible to edge E and intersect the gray-shaded areas that represent the shadow regions for E. Our approach only considers diffraction effects in the shadow regions. These regions are defined as the set of points where only one of the planes that contain the neighboring triangles for E has a normal pointing towards a given point. Edge E must also intersect the shadow regions for each edge $e_{1..10}$ for those edge pairs to be stored in the visibility graph.*

In order to avoid storing reflectance data that is no longer accurate for the current scene configuration, we bound the maximum age in seconds of the data stored in the cache. Any cache entry that is older than some threshold time τ in seconds is removed. This threshold determines the maximum temporal span of the moving average from equations (5) and (6) and the maximum response time for changes in the scene configuration. A larger value for τ increases the accuracy for the estimate of I_i by using a bigger averaging window and more rays. However, this may not be consistent with the current scene configuration if sources, listeners, or objects in the scene change position abruptly. A small value for τ requires more rays to be traced per frame to maintain accurate output, since the temporal averaging for values stored in the cache will have less effect.

Incremental Computation: This diffuse path caching approach incrementally computes a moving average of the incident intensity $I_i(t)$ from equation (2) for each sequence of surface patch reflections that arrive at the listener. We sample these values using traditional path tracing, but use a radiosity-like subdivision to take advantage of the coherence of rays from previous frames and to group the rays based on the sequence of reflections that have occurred. By grouping rays over many frames and reusing those results, we avoid undersampling artifacts, yet need far fewer rays emitted from the sound sources, thereby reducing the time needed to compute realistic diffuse reflections. Like radiosity-based algorithms, our method converges to traditional diffuse path tracing with a suitably small subdivision resolution l, However, if l is too small, it may require a greater number of rays to be traced and a larger diffuse path cache. In this case, fewer rays are grouped together and the effect of path reuse is reduced, resulting in a smaller benefit over traditional diffuse path tracing.

3.2 Edge Visibility Graph for High-Order Diffraction

In order to model edge diffraction, we use an approximation based on the uniform theory of diffraction (UTD), which has been used in interactive geometric sound propagation systems [Tsingos et al. 2001; Taylor et al. 2009; Taylor et al. 2012; Schissler and Manocha 2011]. However, these algorithms are limited to either static scenes or can only compute first order edge diffraction in dynamic scenes.

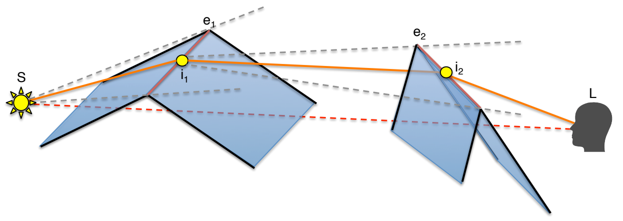

Figure 5: *A second-order diffraction path. Source image positions i_1 and i_2 are only valid if they lie behind the plane formed by the last image position and the current edge. The diffracted sound takes the shortest path over the edges and is valid only if the image positions lie on the edge(s) and if the path is not obstructed.*

The problem of finding high-order diffraction paths efficiently is difficult due to the number of edge pairs that need to be considered. A naive approach has running time that can be exponential in the maximum diffraction order. This is due to the fact that at each level of recursive diffraction, all other diffraction edges in the scene must be considered. Prior methods have used beam tracing [Tsingos et al. 2001] or frustum tracing [Taylor et al. 2009] to compute secondary diffraction edge visibility at runtime. However, this becomes expensive for more than 1st order diffraction in complex scenes, as a large number of complex beam or frustum intersection tests are required.

We present a novel algorithm to compute high-order diffraction paths efficiently using a preprocessed edge visibility graph. This graph structure minimizes the number of diffraction edges that need to be considered at runtime and avoids any runtime edge-edge visibility queries. Most importantly, our approach is valid for any source or listener positions; the visibility graph can be computed once, between all edges of static objects, and then used for all scene configurations. Visibility graphs have been used to accelerate the computation of specular reflections [Funkhouser et al. 1998; Chandak et al. 2009], though our formulation is different.

We compute one visibility graph for all edges of all static objects in the scene. Moreover, a separate visibility graph is computed for the edges of each dynamic object. Our formulation does not, however, take into account the relative visibility of edges of two different dynamic objects or of one static and one dynamic object. Furthermore, we assume that dynamic objects undergo rigid transformations, and that a precomputed visibility graph for that object's static mesh will remain valid. Our formulation allows a simple graph search to be performed at runtime to find high-order diffraction paths that occur within a single graph. We do not consider the visibility between edges belonging to different visibility graphs.

Preprocessing: During the preprocessing step, each edge in a mesh is classified as a diffracting edge or non-diffracting edge based on the angle between the edges' neighboring triangles. We compute a graph data structure containing information about which edges are visible to each of the diffraction edges using region-based visibility algorithms [Antani et al. 2012a]. For each diffraction edge in the mesh, we check all other diffraction edges to see whether they satisfy the orientation criteria for mutually diffracting edges, as shown in Fig. 4. If at least some part of either edge in a pair is behind the plane of the neighboring triangles for the other edge, there is the possibility that diffraction can occur over that edge pair. This test is used to cull edge pairs that cannot be combined as diffraction edges. If two edges can form a diffraction path and are visible to each other, we add a link to the graph structure between these edges. Fig. 4 shows the visible edges for a single edge in an example visibility graph.

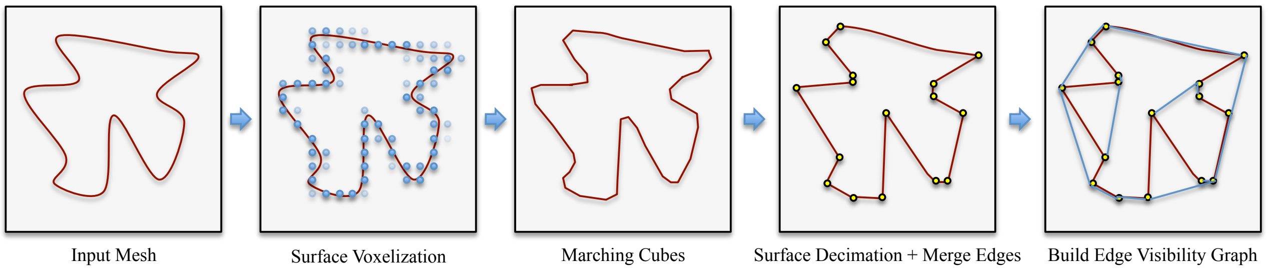

Figure 6: *Stages of our simplification algorithm: surface-voxelization of input mesh; isosurface extractions; surface decimation based on edge-collapses; merge collinear diffraction edges; visibility graph computation*

Runtime Computation: At runtime, our algorithm uses the primary rays traced in the diffuse step to determine a set of triangles visible to each source. For each visible triangle, we check to see if it has any diffraction edges. If so, we search the corresponding visibility graph, moving towards the listener, with that edge as the starting point. The recursive graph search proceeds in a depth-first manner until a maximum depth is reached, at which point the search backtracks and checks other sequences of edges. At each step in the graph search, all diffraction edges that were preprocessed as visible from the current diffraction edge are recursively checked to see if there is a path to the listener. For each edge, we first compute the shortest path between the source and the listener over that edge, then determine the point of closest approach on the edge to the line connecting the source and listener [Economou et al. 2013]. This set of closest points represents the source image positions on each edge in a series of diffractions. A neighboring edge in the graph is checked for higher-order diffraction paths if the point of closest approach on the edge lies on the interval of that edge, and if that point is contained within the previous edge's diffraction shadow region, as shown in Fig. 5. Like previous methods, we only consider diffraction that occurs in the shadow region for an edge [Tsingos et al. 2001]. A point is contained in the shadow region for an edge if it is behind the plane formed by the previous image source position and the edge and is in front of the triangle that defines the shadow region boundary. These simple, fast tests enable us to avoid path validation checks for a large fraction of the potential diffraction paths.

Finally, if the listener is contained within the next diffraction shadow region, we validate the diffraction path to that listener by tracing rays between the previously computed sequence of image positions on the edges. If the ray traced between two consecutive image source positions does not hit an obstacle, that segment of the path is determined to be valid. If the entire path from the source to listener over the sequence of edges is found to be unobstructed, then we compute a frequency-dependent attenuation, using the UTD model, for that path to account for diffraction. Since the UTD attenuation from a single edge diffraction only depends on the local edge geometry and the previous and next source image positions, the attenuation can be computed separately for each edge e_j along a diffraction path. Multiplying the attenuation coefficients for all edges in a path produces the total attenuation due from the high-order diffraction path, similar to the formulation used in [Tsingos et al. 2001]. Each valid path is then added to the final output impulse response for the sound source.

3.3 Voxel-Based Simplification for Edge Diffraction

Many large databases are designed for visual rendering and include highly tessellated models with detailed features. Such models may have higher complexity than that is needed for sound propagation. GA approaches are valid for surfaces that are large compared to the wavelength. There has been some work on simplifying geometric models or use of level-of-detail techniques for acoustic simulation [Siltanen et al. 2008; Pelzer and Vorländer 2010; Tsingos et al. 2007]. However, a key challenge in the field is to automatically generate a simplification that preserves the basic acoustic principles, including reflections, scattering and diffraction. For example, some techniques based on geometric reduction applied to room models can change the reverberation time of the simplified model [Siltanen et al. 2008]. And, in many cases, geometric simplification is performed by hand or using authoring tools, but it is hard to extend these approaches to complex models.

Our approach for computing early reflections and diffractions is based on ray tracing and we use bounding volume hierarchies to accelerate ray intersection tests. In general, the cost of updating the hierarchy for dynamic scenes by refitting is a linear function of the model complexity of dynamic objects. The cost of intersection computation is almost a logarithmic function of the number of polygons. Because of logarithmic complexity, the relative benefit of model simplification on ray-tracing intersection computation is not high. We therefore use the original geometric representation for computing specular and diffuse reflections.

A key aspect of our diffraction algorithm is the identification of important diffraction edges in the scene. The complexity of visibility-graph computation and runtime traversal can increase significantly with the number of edges in the model. Some prior approaches for UTD-based diffraction computation are either limited to coarse models [Tsingos et al. 2001] or consider all edges that have neighboring non-planar triangles [Taylor et al. 2009]. The latter approach can result in large number of small diffraction edges in complex scenes with detailed geometric representations. In practice, the UTD edge diffraction algorithm tends to be more accurate for longer edges; the presence of a high number of small edges can result in inaccurate results.

We present a simplification technique that generates a reduced set of diffraction edges for interactive acoustic simulation. To be specific, we generate meshes corresponding to different simulation wavelengths. Since this simplified mesh is used only for UTD-based edge diffraction computation, the simplification does not affect the the accuracy of reflections. Figure 6 shows an overview of the mesh processing pipeline that we use. This pipeline extends the one described in [Nooruddin and Turk 2003] with additional edge merging and visibility graph steps.

Wavelength-based Simplification: In a preprocessing step, a hierarchical surface voxelization of each object is computed; a voxel's value is determined based on the distance to the closest triangle [Huang et al. 1998]. This allows our method to handle non-closed geometric primitives better than traditional voxelization algorithms, which are based on scan-conversion. The voxelization results in a tree of voxels, where the voxel resolution doubles at each successive tree depth. This tree is used to generate surface approximations corresponding to different wavelengths; we simply choose the tree depth where the voxel resolution is at least half the required wavelength. This resolution is chosen based on the spatial Nyquist distance $h = c/f_{max}$, where f_{max} is the highest simulated frequency [Yeh et al. 2013]. The discretization imposed by the voxelization removes details that are smaller than the voxel resolution.

Next, our approach triangulates a level in the voxel tree by applying the marching cubes algorithm [Lorensen and Cline 1987]. This generates a triangular mesh corresponding to an isosurface in the voxel grid. However, this mesh may not be suitable for computing a reduced set of diffraction edges. For instance, the voxelization and triangulation computation approximate large triangles in the original model with many smaller ones that lie in the same plane. In order to address this issue we first compute the adjacency information for the mesh by merging coincident vertices. Next, we apply the edge-collapse algorithm based on the quadric error metric [Garland and Heckbert 1997] till we exceed an error threshold. These decimation operations progressively merge vertices that share an edge into a single vertex by minimizing the resulting error in the mesh's shape. This results in a highly simplified mesh that preserves the largest features from the original model, while removing small details that would produce extraneous diffraction edges. Finally, we determine a set of candidate diffraction edges using a heuristic that chooses edges with a significant deviation from being planar. More details are given in the appendix. Given this simplified model, we compute the visibility graph and use that for higher order edge diffraction computation.

Reducing Memory Overhead: In order to process very large models efficiently, our algorithm splits the input scene into regions of a maximum size. These regions are voxelized, triangulated, and simplified in parallel. The simplified regions are combined to form the output simplified mesh. The edge collapse algorithm preserves the boundaries of each region in order to avoid seams between them. Since we independently process many smaller regions rather than an entire large mesh at once, the memory footprint of the algorithm is only a few hundred MBs, whereas naively processing an entire large scene could take 10's of GB of RAM.

4 Implementation and Performance

In this section we describe our implementation and highlight its performance on different benchmarks. We also evaluate the error in diffuse reflection computation.

4.1 Implementation

Ray Tracing: We trace rays in a random uniform distribution from each source location to compute diffuse sound. These rays are propagated through the scene via diffuse reflections up to an arbitrary maximum reflection depth (e.g. 10). The number of rays needed to achieve accurate sound is scene-dependent. For all of our benchmarks we traced 1000 rays from each source except where noted. We can use far fewer rays for diffuse sound path tracing than for visual rendering because the listener detection sphere is usually much larger than a camera pixel, and because human hearing is more tolerant of error than visual perception. In addition, the diffuse cache accumulates the results of rays traced on previous frames, thus requiring less rays. Specular reflections are computed separately from diffuse reflections by tracing uniform random rays from the listener's position to sample the set of possible specular paths. We specularly reflect these rays to a chosen maximum depth and use this information to build a set of candidate paths with each path represented as a series of triangle reflectors. Finally, we check each candidate path to see if there is a valid specular reflection along the path from the listener to each source in the scene using the image-source method. If so, an output specular path is produced. This is similar to [Lentz et al. 2007; Schissler and Manocha 2011]. We accelerate ray tracing using bounding volume hierarchies that can be efficiently updated for moving or deforming objects. We use 4-band frequency-dependent reflection attenuation coefficients α that are applied for each material type with the frequency bands: $0-250$ Hz, $250-1000$ Hz, $1000-4000$ Hz, and $4000-22100$ Hz. Each surface material is also assigned a scattering coefficient that determines the fraction of reflected sound that is scattered.

Parallelization: We exploit the SIMD and multi-threading capabilities of current CPUs to accelerate the computation. We run the different components of our sound propagation system separately and in parallel. The diffuse and edge-diffraction components for every sound source are each computed on separate threads that run concurrently. The specular contributions are computed by tracing rays from the listener's position. Once all the threads finish the current frame, the resulting propagation paths for each thread are gathered and sent to the audio rendering subsystem. Our implementation makes use of all available CPU hardware threads. Our sound propagation algorithms are implemented in C++ and make use of SIMD instructions and fast ray tracing. All the timings reported in this paper are measured on a PC with 4-core Intel Core i7 4770K, running at 3.5 GHz with 32 GB of RAM. We use 8 threads on this machine to accelerate the sound computation.

Number of Sources: Our system handles each source separately and the performance changes almost linearly with the number of sources. In Figure 7, we give the breakdown between the different components of our system and how the performance of each component scales with the number of sound sources in large, dynamic outdoor scene with moving sources. In our benchmarks, the diffuse reflections and diffraction dominate the computation, especially for large numbers of sources, since the number of path validation checks scales linearly with the number of sources. This is due to the fact that each source requires many additional rays to be traced. The computation of the specular part is less sensitive to the number of sources because it uses rays traced from the listener rather than each source.

Dynamic Scenes: Our diffuse system supports scenes with moving sound sources, listeners, and objects. The diffuse triangle subdivision (Section 3.1.1) is valid for objects undergoing rigid motion and can be updated in real time if an object deforms or undergoes topological changes. The subdivision can be recomputed for the large city benchmark (254,903 triangles) in 11.5ms using a single CPU core. The bounding volume hierarchy used for ray tracing can also be updated in less than 1ms when objects in a scene undergo rigid motion, and allows fast refitting if objects deform. Since our diffuse technique uses a persistent cache to do time-averaging of diffuse paths, it may also be necessary to clear the cache if there is a large sudden change in the scene. Our diffraction algorithm can also handle moving sources, listeners, and objects, but with only a limited level of dynamism. The high-order diffraction assumes that the visibility relationship between the edges doesn't change. As a result, it doesn't model diffraction effects between the edges of two different dynamic objects or between one dynamic and one static object. However, our approach can model high-order diffraction that occurs between edges of the same dynamic object undergoing affine transformations.

Audio Rendering: In order to render the audio output of our sound propagation algorithms, we use a linearly interpolating delay line for each propagation path [Tsingos et al. 2003]. The smoothness of the interpolation is determined by a parameter that specifies the time for a change in propagation path amplitude or delay. Longer interpolation time produces smoother audio, especially at

Scene	Scene Complexity			Simplification		Visibility Graph		Runtime				
	#Tris	#Edges	Vol.(m^3)	#Edges	Time(s)	Size(MB)	Time(s)	#Sources	#Specular	#Diffuse	#Diffraction	Time(ms)
Sibenik	79942	39082	35.4K	19148	7.90	2.50	6.46	1	10	10	4	22.1
Office	154020	69598	2.2K	4644	10.35	0.32	0.78	6	10	10	5	33.6
Refinery	245828	93046	11.5M	30082	41.52	7.71	23.59	4	10	10	6	27.4
Small City	89792	64853	13.8M	25656	52.55	5.18	31.54	8	10	6	6	19.3
Large City	254903	222680	51.3M	70605	178.5	23.8	299.5	14	10	6	4	71.0

Table 1: *We highlight the model complexity and the performance of simplification and visibility graph computation algorithms. The simplification reduces the number of edges by $3 - 14$ times (shown as simplification-edges). The visibility graph computation is fast and has a small memory overhead. The runtime components include the number of sources in the scene, the maximum order of specular/diffuse reflections and edge diffractions. The frame-time is calculated using 8 threads on a 4-core CPU.*

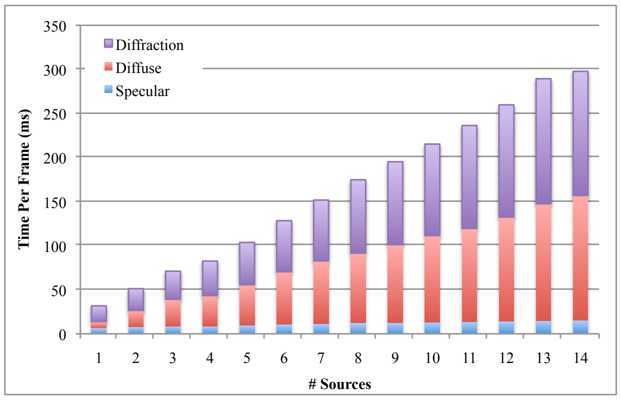

Figure 7: *We highlight the time taken by the different components of our system (specular, diffuse, and diffraction) as the number of sources in the scene increase. All of these times are generated on a single core.*

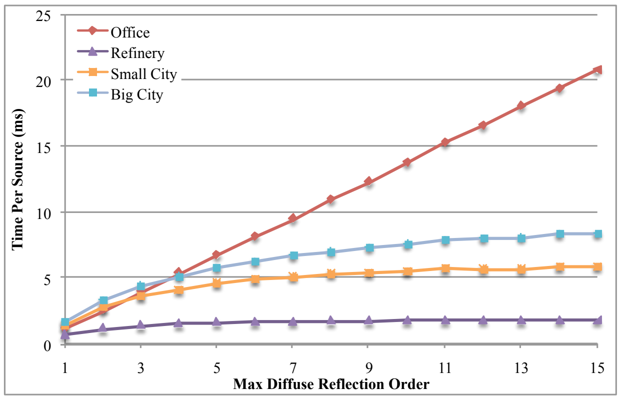

Figure 8: *We highlight how the performance of our diffuse algorithm scales with the maximum diffuse reflection order on a single CPU core. Note that in outdoor scenes, most rays escape the scene after the 4th or 5th bounce. In the indoor office scene, the complexity is linear with respect to the maximum diffuse order.*

the boundary between the lit and diffraction shadow region, but results in a higher latency for these transitions. The source audio is split at runtime into 4 frequency bands that correspond to the bands used for material properties with Linkwitz-Riley 4th-order crossover filters. This allows our renderer to efficiently model frequency-dependent effects by applying different gains to each band. Audio for all frequency bands is rendered separately based on the frequency-dependent attenuation coefficients for the path, then mixed (added) together at the output to produce the final audio. We perform vector-based amplitude panning to spatialize the audio for each propagation path separately using the path's direction from the listener. As the audio for each path is rendered, it is accumulated in a common output audio buffer. We use a statistical model for late-reverberation based on the Eyring reverb time equation [Eyring 1930] that dynamically estimates the mean free path and visible surface area in the scene using diffuse sound rays. The mean free path is used to approximate the effective scene volume with the well-known equation $V = \bar{l}S/4$, relating the volume (V), mean free path ($\bar{l}$), and surface area (S) of the environment: The RT_{60} from this model is used as input for a Schroeder-type reverberator [Schroeder 1962] that is mixed with the early propagation paths computed by our GA algorithm. A delay based on the shortest diffuse path delay time is applied to the reverb to align it with the early diffuse reflections. All audio rendering is performed at 44.1 kHz and we use SIMD instructions to vectorize the rendering of frequency bands.

4.2 Benchmarks

We evaluated our sound propagation system on indoor and outdoor scenes with large volumes and high model complexity. Our approach can generate plausible acoustic effects, including specular and diffuse reflections and edge-diffraction at interactive rates on a 4-core CPU (see Table 1).

Office: The listener walks through a moderate-sized office environment with six sound sources. We demonstrate fourth order diffraction effects around corners that produce low-pass filtering for occluded sources. Diffuse reflections are a significant component of this indoor scene due to material properties with high scattering coefficients.

Refinery: This large outdoor scene of an oil refinery demonstrates the need for high-order diffraction computation, as the sound waves diffract around curved obstacles like oil tanks and smoke stacks. A helicopter flies through the scene and passes behind these obstacles, demonstrating the performance of our high-order diffraction. Our simplification scheme removes the small details from the original mesh, resulting in a smaller set of diffraction edges.

Small City: This small city scene demonstrates that our system can handle both moving sources, listeners, and obstacles. The listener sits in a vehicle that drives through a city scene, where it passes other cars that are moving sound sources as well as obstacles.

Big City: A pedestrian listener walks along the sidewalk at an intersection in a large city model with more than 50 buildings over an area of 0.5 km^2. There are 14 moving sound sources (cars, ambulance) that pass by the pedestrian listener. These vehicles are also moving obstacles. This scene shows how our approach can scale well with the number of sources and maintain interactive frame rates for large scenes.

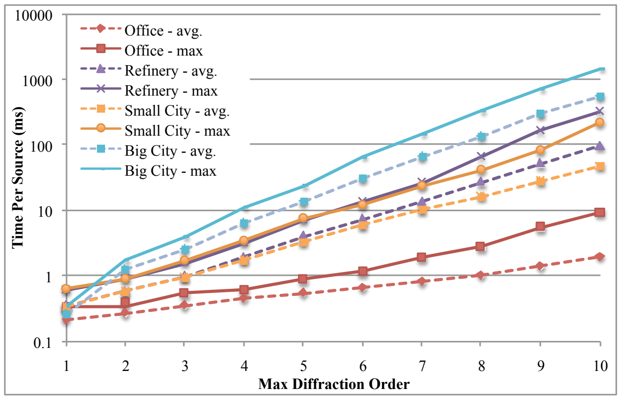

Figure 9: *We show the performance of our graph-based diffraction algorithm up to 10th order for 4 benchmark scenes. Since the performance varies as a function of the source and listener locations, we show the average time spent in edge diffraction as well as the maximum time. Please note the logarithmic scale on the time axis.*

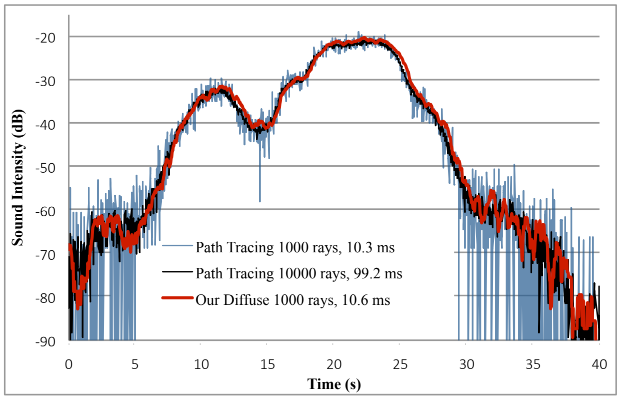

Figure 10: *We compare the sound intensity received at the listener computed by our diffuse reflection method with that of the traditional path tracing algorithm in the office benchmark. We compute reflections up to 10th order and average the sound intensity over the frequency bands described in Section 4.1. The accuracy of our algorithm (with 1000 rays) is comparable to the path tracing algorithm (using 10000 rays). On the other hand, path tracing with 1000 rays misses many contributions and results in inaccurate diffuse contributions.*

4.3 Diffuse Reflections

We have analyzed the runtime performance as well as accuracy of our diffuse reflection computation algorithms. We chose to use a value of $l = 0.5m$ for our simulations. We give more details on the accuracy of the algorithm as a function of patch size (l) in the appendix. Fig. 8 shows that the performance of our algorithm scales for increasing maximum diffuse reflection order. In practice, our incremental algorithm is able to simulate over 10 orders of reflection in the scenes at around $50-60$Hz for a single sound source. We also compared the accuracy of our algorithm with traditional path tracing (see Fig. 10). While we use 1000 rays with our approach, we compare its accuracy with two versions of path tracing: 1000 rays and 10000 rays, and perform 10 orders of reflection. The accuracy of our algorithm is comparable to that of path tracing with 10000 rays, with an average error of of 2.27 dB. On the other hand, path tracing with only 1000 rays, results in noisier results and average error of 6.69 dB. The temporal averaging of our method dramatically improves the results for a given number of emitted rays (i.e. 1000 rays). Our approach is effective at improving the accuracy of low-intensity sound in the left and right portions of the graph. Our supplementary video includes audio samples that compare path tracing with our approach.

4.4 Edge Diffraction

In order to evaluate the performance of high order edge diffraction algorithm, we measured how our approach scales with the maximum diffraction order in Fig 9. In the worst case, the complexity of GA-based diffraction algorithms is of the form $O(n^d)$ where n is the number of neighbors for each edge in the visibility graph and d is the maximum diffraction order. We report both the average time to compute diffraction for the benchmark scenes, as well as the maximum time spent for any instance of the source and location. This is due to the fact that the performance of our diffraction varies considerably with the source and listener positions. For example, for certain positions, the time spent in searching the visibility graph can be high, as some of the vertices in the visibility graph may have a high number of neighbors. In practice, our approach enables computation of 5th or 6th order diffraction at real-time rates in our benchmarks. Since we use precomputed visibility information, no runtime edge-edge visibility checks are performed. This dramatically reduces the number of edge pairs that need to be considered for high-order diffraction paths.

4.5 Simplification

Our simplification algorithm can generate different approximations as a function of the wavelength (see the appendix for details). In our implementation, we generated the simplifications based on wavelength $\lambda = 0.25$m, corresponding to a frequency of 1.3kHz, and a voxel size of 0.125m. We found that our simplification algorithm significantly reduces the number of diffraction edges for the benchmark scenes. Table 1 shows that the number of edges is reduced to around $30 - 90\%$ of the original number of diffraction edges for the unsimplified model. For small scenes, the simplification algorithm takes only a few seconds while large scenes that are as large as 50 million m^3 can be simplified in minutes. In general, the simplification time increases with the scene volume because more voxels are needed to meet the wavelength spatial resolution. The voxelization approach is $O(n \log n)$ with respect to the number of triangles in original mesh. We use simplified models for visibility graph computation. Since we reduce the number of edges, it significantly speeds up visibility graph computation and also reduces the size of visibility graph.

5 Comparison with Previous Works

In this section we compare the results of our algorithms with prior techniques. This includes diffuse reflections, edge diffraction, as well as interactive geometric propagation systems.

The prior geometric techniques for diffuse reflections are based on path tracing [Lentz et al. 2007; Alarcao et al. 2009; Taylor et al. 2009]. We have compared the accuracy as well as runtime performance with path tracing algorithms in Fig. 10. The main benefit of our method arises from the fact that we can to shoot almost one order of magnitude fewer rays as compared to path tracing to achieve similar accuracy. This is due to the fact that we perform temporal averaging that can significantly improve the accuracy. The RESound system [Taylor et al. 2009] takes about $250-500$ms to compute up to 3 orders of diffuse reflections (with 200K rays) on models with $60 - 280$K triangles using seven threads on a multi-core

CPU. On the other, our algorithm takes less than 15ms per source to compute up to 10 orders of diffuse reflections. Other recent work is based on the acoustic rendering equation [Siltanen et al. 2007; Antani et al. 2012b] and is used to precompute higher order reflections and diffraction for mostly static scenes. These approaches are complimentary to our formulation. For example, our diffuse algorithm can be used to accelerate early reflection computation in [Antani et al. 2012b].

In terms of edge diffraction, prior techniques are limited to coarse static models [Tsingos et al. 2001] or first order edge diffraction in dynamic scenes [Taylor et al. 2012; Schissler and Manocha 2011]. These approaches make no assumptions on edge visibility at runtime and therefore must compute a visible set of high-order diffraction edges for each edge on every frame. Generally this operation is performed by intersecting shadow-region frusta with the scene or by sampling edge visibility by tracing rays in the shadow region. This must be performed recursively for each edge considered for diffraction and becomes non-interactive (i.e. more than $500-1000$ ms) at more than one or two orders of diffraction. Furthermore, we use wavelength-based simplification, which makes it possible to perform high-order edge diffraction in complex scenes.

We compared our UTD-based diffraction technique with the offline BTM diffraction model [Svensson et al. 1999] on a simple scene with a rectangular obstacle (12 edges) and a single sound source. The BTM model integrates the diffraction that occurs over the entire extent of each edge, whereas UTD only considers diffraction over a single point on an edge. Figure 11 summarizes the results of this comparison for 3 different listener locations corresponding to direct sound, 1st order diffraction, and 2nd order diffraction. We observed that our formulation based on UTD diffraction model overestimates the amount of high-frequency attenuation versus BTM. The error in the frequency response was 3.10dB for 1st-order diffraction and 3.61dB for 2nd-order diffraction. We have also performed an audio comparison of our approach and the BTM approach and included the results in the supplementary video.

In terms of overall system comparisons, our geometric propagation system has many additional capabilities and improved performance as compared to prior approaches that are based on ray tracing and can handle dynamic scenes. The RAVEN system [Lentz et al. 2007; Pelzer and Vorländer 2010] is a state of the art interactive geometric sound propagation system. It is mainly designed of indoor scenes and supports specular reflections based on image source method, diffuse reflections based on path tracing, diffraction, and also uses simplified models. Our underlying diffuse reflection algorithm is significantly faster. Plus we can perform efficient high-order diffraction and also handle complex outdoor scenes. The RESound system [Taylor et al. 2009] only supports first-order edge diffraction and our diffuse reflection computation is significantly faster. The guided multi-view ray tracing algorithm [Taylor et al. 2012] can be used to choose the ray-budget in our approach. Furthermore, that algorithm uses a parallel GPU-based ray tracer to accelerate the computations. It takes about 93 ms using a CPU and a GPU to compute three orders of specular reflections and one order of edge diffraction on the Sibenik model. On the other hand, our approach can handle models with higher complexity with ten orders of specular and diffuse reflections and five orders edge diffraction in $27-33$ms on a 4-core CPU.

6 Conclusions, Limitations, and Future Work

We have presented different algorithms to enable interactive geometric sound propagation in complex scenes. Our main contributions include a novel algorithm for diffuse reflections and higher order diffraction. We also present an approach to simplify the scene

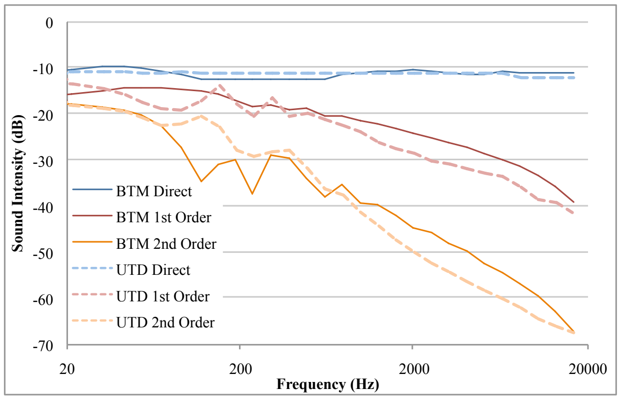

Figure 11: *This graph compares the 1/3 octave frequency response of our diffraction approach to that of the more-accurate offline BTM method for 3 different listener locations corresponding to direct sound, 1st-order diffraction, and 2nd-order diffraction. We observe that the UTD diffraction model overestimates the attenuation of high frequencies. The average errors for the 3 listeners were 0.67dB, 3.10dB, and 3.61dB, respectively.*

for edge diffraction and thereby making it possible to automatically handle large geometric databases for sound propagation. We have highlighted the performance on many large and complex dynamic scenes. We observe more than an order-of-magnitude performance improvement over prior methods and the accuracy is comparable to those methods. To the best of our knowledge, this is the first approach that can interactively compute higher-order diffraction and diffuse reflections in complex environments to generate plausible sound effects.

Limitations: Our approach is based on ray tracing, and therefore all standard limitations of GA are inherent to our formulation. These include potential inaccuracies for low frequencies where we assume that primitives are larger than the wavelength. Our approach assumes that the medium is homogeneous and doesn't model any changes in temperature or pressure in large outdoor scenes. Since we model all sound sources as point sources, the audio may not match the visual representations of large objects such as the helicopter shown in the supplementary video. We compute specular, diffuse, and diffraction sound components separately, then combine them for audio rendering. This approach loses some energy because combinations of diffraction and reflection are not computed.

Our diffuse reflection algorithm can have some error as compared to path tracing methods, especially if there is a large relative change in the position of sources, objects, or the listener between two successive frames. In this case, the diffuse cache can be reset to avoid any errors in the diffuse component of the sound generated for future frames The subdivision size used may also cause error in the delay time or the direction of propagation paths.

As discussed in Section 4.1, our high-order diffraction algorithm only supports a limited level of dynamism due to the use of precomputed edge visibility.

It is possible that our simplification algorithm can introduce inaccuracies in diffraction computation, since large voxel sizes may produce significant error in the resulting diffraction edges. This can cause discontinuities in the transition between direct and diffracted sound at the shadow region boundary. To reduce this error, we can choose a small voxel size for simplification and use slower interpolation for audio rendering as discussed in Section 4.1. Furthermore, the simplified representation for edge diffraction is stored in addition to the original model representation (and hierarchy) for reflections. This moderately increases the storage overhead.

In the future, we would like to apply the more accurate, but also more expensive, BTM diffraction model to our high-order diffraction path finding. We would also like to further extend our simplification algorithm to handle reflections as well as material properties. We would like to extend to heterogeneous environments using non-linear ray tracing. It would be useful to combine with perceptual techniques [Moeck et al. 2007] to handle large number of sources and accelerate sound rendering. Furthermore, we would like to evaluate the benefits of our approach based on user-studies.

Acknowledgements

We would like to thank the anonymous reviewers for comments and feedback. We would also like to thank Valve Corporation for permission to use Half-Life 2 sound effects in the Office and Refinery scenes. This research was supported in part by the Link Foundation Fellowship in Advanced Simulation and Training, ARO Contracts W911NF-10-1-0506, W911NF-12-1-0430, W911NF-13-C-0037, and the National Science Foundation (NSF awards 0917040, 1320644),

References

ALARCAO, D., SANTOS, D., AND COELHO, J. L. B. 2009. An auralization system for real time room acoustics simulation. In *Proceedings of Tecniacustica*.

ALLEN, J. B., AND BERKLEY, D. A. 1979. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America 65*, 4 (April), 943–950.

ANTANI, L., AND MANOCHA, D. 2013. Aural proxies and directionally-varying reverberation for interactive sound propagation in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on 19*, 4, 567–575.

ANTANI, L., CHANDAK, A., TAYLOR, M., AND MANOCHA, D. 2012. Efficient finite-edge diffraction using conservative from-region visibility. *Applied Acoustics 73*, 218–233.

ANTANI, L., CHANDAK, A., SAVIOJA, L., AND MANOCHA, D. 2012. Interactive sound propagation using compact acoustic transfer operators. *ACM Trans. Graph. 31*, 1 (Feb.), 7:1–7:12.

ATTENBOROUGH, K., LI, K. M., AND HOROSHENKOV, K. 2007. *Predicting Outdoor Sound*. Taylor and Francis, New York.

BERTRAM, M., DEINES, E., MOHRING, J., JEGOROVS, J., AND HAGEN, H. 2005. Phonon tracing for auralization and visualization of sound. In *Proceedings of IEEE Visualization*, 151–158.

BORISH, J. 1984. Extension to the image model to arbitrary polyhedra. *The Journal of the Acoustical Society of America 75*, 6 (June), 1827–1836.

CHANDAK, A., ANTANI, L., TAYLOR, M., AND MANOCHA, D. 2009. Fastv: From-point visibility culling on complex models. *Computer Graphics Forum (Proc. of EGSR) 28*, 3, 1237–1247.

DROSS, P., SCHRÖDER, D., AND VORLÄNDER, M. 2007. A fast reverberation estimator for virtual environments. In *Audio Engineering Society Conference: 30th International Conference: Intelligent Audio Environments*, Audio Engineering Society.

ECONOMOU, P., CHARALAMPOUS, P., IOANNIDES, S., AND POLYKARPOU, P. 2013. The significance of sound diffraction effects in predicting acoustics in ancient theatres. *Acta Acustica united with Acustica 99*, 1, 48–57.

EMBRECHTS, J. J. 2000. Broad spectrum diffusion model for room acoustics ray-tracing algorithms. *The Journal of the Acoustical Society of America 107*, 4, 2068–2081.

EYRING, C. F. 1930. Reverberation time in "dead" rooms. *The Journal of the Acoustical Society of America 1*, 2A (January), 217–241.

FRANZONI, L. P., BLISS, D. B., AND ROUSE, J. W. 2001. An acoustic boundary element method based on energy and intensity variables for prediction of high-frequency broadband sound fields. *The Journal of the Acoustical Society of America 110*, 3071.

FUNKHOUSER, T., CARLBOM, I., ELKO, G., PINGALI, G., SONDHI, M., AND WEST, J. 1998. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proc. of ACM SIGGRAPH*, 21–32.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 209–216.

GUMEROV, N. A., AND DURAISWAMI, R. 2009. A broadband fast multipole accelerated boundary element method for the three-dimensional helmholtz equation. *J. Acoustical Society of America 125*, 1, 191–205.

HUANG, J., YAGEL, R., FILIPPOV, V., AND KURZION, Y. 1998. An accurate method for voxelizing polygon meshes. In *Volume Visualization, 1998. IEEE Symposium on*, IEEE, 119–126.

JAMES, D. L., BARBIC, J., AND PAI, D. K. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. In *Proc. of ACM SIGGRAPH*, 987–995.

KOUYOUMJIAN, R. G., AND PATHAK, P. H. 1974. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proceedings of the IEEE 62*, 11, 1448–1461.

KRIVANEK, J., GAUTRON, P., WARD, G., JENSEN, H., TABELLION, E., AND CHRISTENSEN, P. 2008. *Practical Global Illumination with Irradiance Caching*. ACM SIGGRAPH Course Notes.

KROKSTAD, A., STROM, S., AND SORSDAL, S. 1968. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibration 8*, 1 (July), 118–125.

KUTTRUFF, H. 1995. A simple iteration scheme for the computation of decay constants in enclosures with diffusely reflecting boundaries. *The Journal of the Acoustical Society of America 98*, 1, 288–293.

KUTTRUFF, H. 2007. *Acoustics: An Introduction*. Taylor and Francis, New York.

LENTZ, T., SCHRÖDER, D., VORLÄNDER, M., AND ASSENMACHER, I. 2007. Virtual reality system with integrated sound field simulation and reproduction. *EURASIP Journal on Advances in Singal Processing 2007* (January), 187–187.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, vol. 21, ACM, 163–169.

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. on Graphics 32*, 2, 19:1–19:13.

MEHRA, R., ANTANI, L., KIM, S., AND MANOCHA, D. 2014. Source and listener directivity for interactive wave-based sound propagation. *Visualization and Computer Graphics, IEEE Transactions on 20*, 4, 495–503.

MOECK, T., BONNEEL, N., TSINGOS, N., DRETTAKIS, G., VIAUD-DELMON, I., AND ALLOZA, D. 2007. Progressive perceptual audio rendering of complex scenes. In *Proceedings of Symposium on Interactive 3D graphics and games*, ACM, 189–196.

NIRONEN, H. 2004. *Diffuse Reflections in Room Acoustics Modelling*. PhD thesis, Helsinki University of Technology.

NOORUDDIN, F. S., AND TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *Visualization and Computer Graphics, IEEE Transactions on 9*, 2, 191–205.

PELZER, S., AND VORLÄNDER, M. 2010. Frequency-and time-dependent geometry for real-time auralizations. In *Proceedings of 20th International Congress on Acoustics, ICA*.

RAGHUVANSHI, N., SNYDER, J., MEHRA, R., LIN, M., AND GOVINDARAJU, N. 2010. Precomputed wave simualtion for real-time sound propagation of dynamic sources in complex scenes. *ACM Trans. on Graphics 29*, 4, 68:1 – 68:11.

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)*.

SCHISSLER, C., AND MANOCHA, D. 2011. Gsound: Interactive sound propagation for games. In *AES 41st International Conference: Audio for Games*.

SCHROEDER, M. R. 1962. Natural sounding artificial reverberation. *Journal of the Audio Engineering Society 10*, 3, 219–223.

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 (September), 1624–1635.

SILTANEN, S., LOKKI, T., SAVIOJA, L., AND LYNGE CHRISTENSEN, C. 2008. Geometry reduction in room acoustics modeling. *Acta Acustica united with Acustica 94*, 3, 410–418.

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

TAYLOR, M., CHANDAK, A., ANTANI, L., AND MANOCHA, D. 2009. Resound: interactive sound rendering for dynamic virtual environments. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, ACM, 271–280.

TAYLOR, M., CHANDAK, A., MO, Q., LAUTERBACH, C., SCHISSLER, C., AND MANOCHA, D. 2012. Guided multiview ray tracing for fast auralization. *IEEE Transactions on Visualization and Computer Graphics 18*, 1797–1810.

THOMPSON, L. L. 2006. A review of finite-element methods for time-harmonic acoustics. *J. Acoustical Society of America 119*, 3, 1315–1330.

TSINGOS, N., FUNKHOUSER, T., NGAN, A., AND CARLBOM, I. 2001. Modeling acoustics in virtual environments using the uniform theory of diffraction. In *SIGGRAPH 2001, Computer Graphics Proceedings*, 545–552.

TSINGOS, N., GALLO, E., AND DRETTAKIS, G. 2003. Perceptual audio rendering of complex virtual environments. Tech. Rep. RR-4734, INRIA, REVES/INRIA Sophia-Antipolis, Feb.

TSINGOS, N., DACHSBACHER, C., LEFEBVRE, S., AND DELLEPIANE, M. 2007. Instant sound scattering. In *Proceedings of the Eurographics Symposium on Rendering*, 111–120.

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

VORLÄNDER, M. 1989. Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm. *The Journal of the Acoustical Society of America 86*, 1, 172–178.

WAND, M., AND STRASSER, W. 2004. Multi-resolution sound rendering. In *SPBG'04 Symposium on Point - Based Graphics 2004*, 3–11.

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, 165:1–165:11.