A beam tracing method for interactive architectural acoustics

Thomas Funkhouser[a)]
Princeton University, Princeton, New Jersey 08540

Nicolas Tsingos
INRIA, Sophia-Antipolis, France

Ingrid Carlbom
Lucent Bell Laboratories, Murray Hill, New Jersey

Gary Elko and Mohan Sondhi
Avaya Labs, Basking Ridge, New Jersey 07920

James E. West
The John Hopkins University, Baltimore, Maryland 21218

Gopal Pingali
IBM TJ Watson Research Center, Hawthorne, New York 10532

Patrick Min and Addy Ngan
Princeton University, Princeton, New Jersey 08540

(Received 9 May 2002; revised 11 April 2003; accepted 25 August 2003)

A difficult challenge in geometrical acoustic modeling is computing propagation paths from sound sources to receivers fast enough for interactive applications. This paper describes a beam tracing method that enables interactive updates of propagation paths from a stationary source to a moving receiver in large building interiors. During a precomputation phase, convex polyhedral beams traced from the location of each sound source are stored in a ''beam tree'' representing the regions of space reachable by potential sequences of transmissions, diffractions, and specular reflections at surfaces of a 3D polygonal model. Then, during an interactive phase, the precomputed beam tree(s) are used to generate propagation paths from the source(s) to any receiver location at interactive rates. The key features of this beam tracing method are (1) it scales to support large building environments, (2) it models propagation due to edge diffraction, (3) it finds all propagation paths up to a given termination criterion without exhaustive search or risk of under-sampling, and (4) it updates propagation paths at interactive rates. The method has been demonstrated to work effectively in interactive acoustic design and virtual walkthrough applications. © *2004 Acoustical Society of America.* [DOI: 10.1121/1.1641020]

PACS numbers: 43.55.Ka, 43.58.Ta [VWS] Pages: 739–756

I. INTRODUCTION

Geometric acoustic modeling tools are commonly used for design and simulation of 3D architectural environments. For example, architects use CAD tools to evaluate the acoustic properties of proposed auditorium designs, factory planners predict the sound levels at different positions on factory floors, and audio engineers optimize arrangements of loudspeakers. Acoustic modeling can also be useful for providing spatialized sound effects in interactive virtual environment systems.[1,2]

One major challenge in geometric acoustic modeling is accurate and efficient computation of propagation paths.[3] As sound travels from source to receiver via a multitude of paths containing reflections, transmissions, and diffractions (see Fig. 1), accurate simulation is extremely compute intensive. Most prior systems for geometric acoustic modeling have been based on image source methods[4,5] and/or ray tracing,[6] and therefore they do not generally scale well to support large 3D environments, and/or they fail to find all significant propagation paths containing edge diffractions. These systems generally execute in ''batch'' mode, taking several seconds or minutes to update the acoustic model for a change of the source location, receiver location, or acoustical properites of the environment,[7] and they allow visual inspection of propagation paths only for a small set of prespecified source and receiver locations.

In this paper, we describe a beam tracing method that computes early propagation paths incorporating specular reflection, transmission, and edge diffraction in large building interiors fast enough to be used for interactive applications. While different aspects of this method have appeared at computer graphics conferences,[8–10] this paper provides the first complete description of the proposed acoustic modeling system.

Briefly, our system executes as follows. During an offline precomputation, we construct a spatial subdivision in which 3D space is partitioned into convex polyhedra (cells). Later, for each sound source, we trace beams through the spatial subdivision constructing a ''beam tree'' data structure

a)Electronic mail: funk@cs.princeton.edu

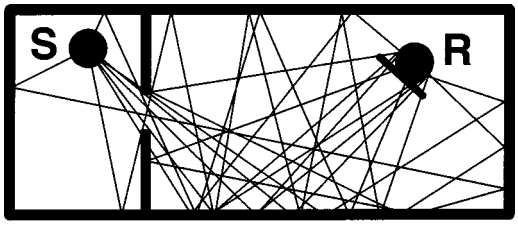


FIG. 1. Propagation paths.

encoding convex polyhedral regions of space reachable from the source by different sequences of scattering events. Then, during an interactive session, the beam trees are used to find propagation paths from the source and an arbitrary receiver location. The updates for each receiver are quick enough to be applied in an interactive acoustic design application. (For simplicity of exposition, our paper considers only propagation paths traced from sound sources to receivers. However, paths from receivers to sources could be computed just as easily—simply switch the terms ''source'' and ''receiver'' in the following text.)

The most important contribution of this paper is a method for precomputing data structures that encode potential sequences of surface scattering in a manner that enables interactive updates of propagation paths from a stationary source location to an arbitrarily moving receiver location. Our algorithms for construction and query of these data structures have the unique features that they scale well with increasing geometric complexity in densely occluded environments and that they generate propagation paths with any combination of transmission, specular reflection, and diffraction without risk of undersampling. We have incorporated these data structures and algorithms into a system that supports real-time auralization and visualization of large virtual environments.

The remainder of the paper is organized as follows. The next section reviews previous work in geometric acoustic modeling. Section III contains an overview of our system, with details of the spatial subdivision, beam tracing, path generation, and auralization methods appearing in Sec. IV. Section V contains experimental results. Applications, limitations, and topics for future work are discussed in Sec. VI. Finally, Sec. VII contains a brief conclusion.

II. PREVIOUS WORK

There have been decades of work in acoustic modeling of architectural environments, including several commercial systems for computer-aided design of concert halls (e.g., Refs. 11–13). Surveys can be found in Refs. 3 and 7.

Briefly, prior methods can be classified into two major types: (1) numerical solutions to the wave equation using finite/boundary element methods (FEM/BEM) and (2) high-frequency approximations based on geometrical propagation paths. In the latter case, image source methods, ray tracing, and beam tracing have been used to construct the sound propagation paths.

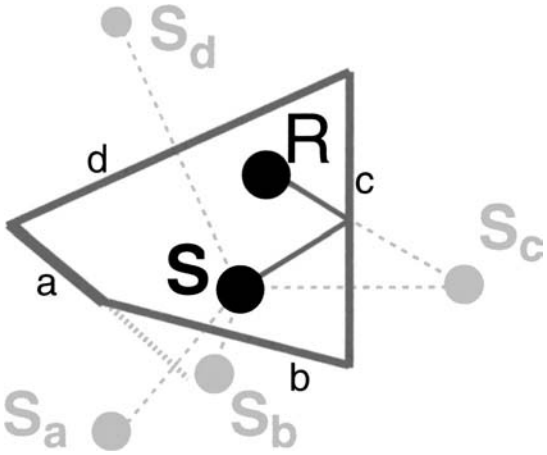


FIG. 2. Image source method.

A. Boundary element methods

Finite and boundary element methods solve the wave equation (and associated boundary conditions), subdividing space (and possibly time) into *elements*.[14–17] The wave equation is then expressed as a discrete set of linear equations for these elements. The boundary integral form of the wave equation (i.e., Green's or Helmoltz–Kirchhoff's equation) can be solved by subdividing only the boundaries of the environment and assuming the pressure (or particle velocity) is a linear combination of a finite number of basis functions on the elements. One can either impose that the wave equation is satisfied at a set of discrete points (collocation method) or ensure a global convergence criteria (Galerkin method). In the limit, finite element techniques provide an accurate solution to the wave equation. However, they are mainly used at low frequencies and for simple environments since the compute time and storage space increase dramatically with frequency.

Finite element techniques are also used to model *energy* transfer between surfaces. Such techniques have already been applied in acoustics,[18,19] as well as other fields,[20,21] and provide an efficient way of modeling diffuse global energy exchanges (i.e., where surfaces are lambertian reflectors). While they are well suited for computing energy decay characteristics in a given environment, energy exchange techniques do not allow direct reconstruction of an impulse response. Instead, they require the use of an underlying statistical model and a random phase assumption.[22] Moreover, most surfaces act primarily as specular or glossy reflectors for sound. Although extensions to nondiffuse environments have been proposed in computer graphics,[21,20] they are often time and memory consuming and not well suited to interactive applications.

B. Image source methods

Image source methods[4,5] compute specular reflection paths by considering *virtual sources* generated by mirroring the location of the audio source, *S*, over each polygonal surface of the environment (see Fig. 2). For each virtual source, S_i, a specular reflection path can be constructed by iterative intersection of a line segment from the source position to the

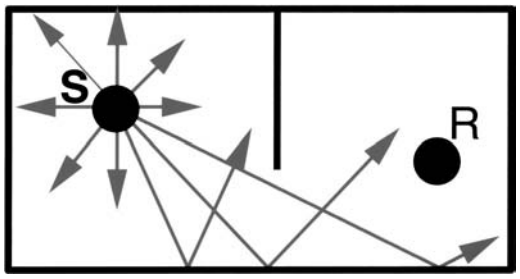


FIG. 3. Ray tracing method.

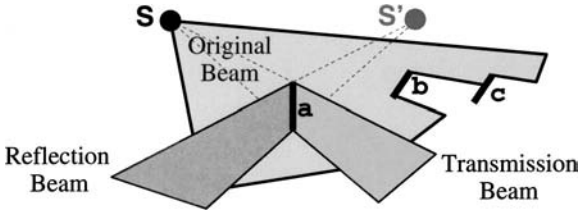


FIG. 4. Beam tracing method.

receiver position, R, with the reflecting surface planes (such a path is shown for virtual source S_c in Fig. 2). Specular reflection paths are computed up to any order by recursive generation of virtual sources.

The primary advantage of image source methods is their robustness. They guarantee that all specular paths up to a given order or reverberation time are found. However, the basic image source method models only specular reflection, and their expected computational complexity grows exponentially. In general, $O(n^r)$ virtual sources must be generated for r reflections in environments with n surface planes. Moreover, in all but the simplest environments (e.g., a box), complex validity/visibility checks must be performed for each of the $O(n^r)$ virtual sources since not all of the virtual sources represent physically realizable specular reflection paths.[5] For instance, a virtual source generated by reflection over the nonreflective side of a surface is "invalid."[5] Likewise, a virtual source whose reflection is blocked by another surface in the environment or intersects a point on a surface's plane which is outside the surface's boundary (e.g., S_a in Fig. 2) is "invisible."[5] During recursive generation of virtual sources, descendents of invalid virtual sources can be ignored. However, descendents of invisible virtual sources must still be considered, as higher-order reflections may generate visible virtual sources (consider mirroring S_a over surface d). Due to the computational demands of $O(n^r)$ visibility checks, image source methods are practical for modeling only a few specular reflections in simple environments.[23]

C. Ray tracing methods

Ray tracing methods[6] find propagation paths between a source and receiver by generating rays emanating from the source position and following them through the environment until a set of rays has been found that reach the receiver (see Fig. 3).

The primary advantage of these methods is their simplicity. They depend only on ray–surface intersection calculations, which are relatively easy to implement and have computational complexity that grows sublinearly with the number of surfaces in the model. Another advantage is generality. As each ray–surface intersection is found, paths of specular reflection, diffuse reflection, diffraction, and refraction can be sampled,[24,25] thereby modeling arbitrary types of propagation, even for models with curved surfaces. The primary disadvantages of ray tracing methods stem from their discrete sampling of rays, which may lead to undersampling errors in predicted room responses.[26] For instance, the receiver position and diffracting edges are often approximated by volumes of space (in order to enable intersections with infinitely thin rays), which can lead to false hits and paths counted multiple times.[26] Moreover, important propagation paths may be missed by all samples. In order to minimize the likelihood of large errors, ray tracing systems often generate a large number of samples, which requires a large amount of computation. Another disadvantage of ray tracing is that the results are dependent on a particular receiver position, and thus these methods are not directly applicable in interactive applications where either the source or receiver can move.

D. Beam tracing methods

Beam tracing methods[27,28] classify propagation paths from a source by recursively tracing pyramidal beams (i.e., sets of rays) through the environment (see Fig. 4). Briefly, for each beam, polygons in the environment are considered for intersection with the beam in front-to-back visibility order (i.e., such that no polygon is considered until all others that at least partially occlude it have already been considered). As intersecting polygons are detected, the original beam is clipped to remove the shadow region, a transmission beam is constructed matching the shadow region, and a reflection beam is constructed by mirroring the transmission beam over the polygon's plane. This method has been used in a variety of applications, including acoustic modeling,[27,8,29–31] illumination,[32–35,28,36] visibility determination,[37–39] and radio propagation prediction.[40,41]

The primary advantage of beam tracing is that it leverages geometric coherence, since each beam represents an infinite number of potential ray paths emanating from the source location. It does not suffer from the sampling artifacts of ray tracing,[26] nor the overlap problems of cone tracing,[42,43] since the entire space of directions leaving the source can be covered by beams exactly. The disadvantage is that the geometric operations required to trace beams through a 3D model (i.e., intersection and clipping) are relatively complex, as each beam may be reflected and/or obstructed by several surfaces.

Some systems avoid the geometric complexity of beam tracing by approximating each beam by its medial axis ray for intersection and mirror operations,[44] possibly splitting rays as they diverge with distance.[45,46] In this case, the beam representation is useful only for modeling the distribution of rays/energy with distance and for avoiding large tolerances in ray–receiver intersection calculations. If beams are not clipped or split when they intersect more than one surface, significant propagation paths can be missed, and the computed acoustical field can be grossly approximated.

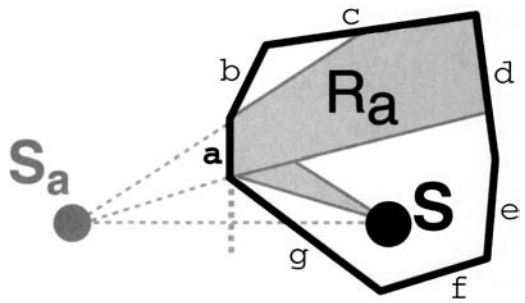


FIG. 5. Beam tracing culls invisible virtual sources.

III. OVERVIEW OF APPROACH

Our approach is based on polyhedral beam tracing. The strategy is to trace beams that decompose the space of rays into topologically distinct bundles corresponding to potential sequences of scattering events at surfaces of the 3D scene (*propagation sequences*), and then use them to guide efficient generation of propagation paths between a sound source and receiver in a later interactive phase. This approach has several advantages:

(i) **Efficient enumeration of propagation sequences**: Beam tracing provides a method for enumerating potential sequences of surface scattering events *without exhaustive search*, as in image source methods.[4,5] Since each beam describes the region of space containing all possible rays representing a particular sequence of scattering events, only surfaces intersecting the beam must be considered for further propagation. For instance, in Fig. 5, consider the virtual source S_a, which results from mirroring S over polygon a. The corresponding specular reflection beam, R_a, contains exactly the set of receiver points for which S_a is valid and visible. Similarly, R_a intersects exactly the set of polygons (c and d) for which second-order reflections are possible after specular reflection off polygon a. Other polygons (b, e, f, and g) need not be considered for higher-order propagation after specular reflection off a. As in this example, beam tracing can be used to prune the combinatorial search space of propagation sequences without resorting to sampling.

(ii) **Deterministic computation**: Beam tracing provides a method for finding potential sequences of diffracting edges and reflecting faces *without risk of errors due to under-sampling*, as in ray tracing. Since the entire 2D space of directions leaving the source can be partitioned so that every ray is in exactly one beam, beam tracing methods can guarantee finding every propagation path up to a specified termination criteria. Moreover, beams support well-defined intersections with points and edges, and thus beam tracing methods do not generate the systematic errors of ray tracing due to approximations made in intersecting infinitely thin rays with infinitely thin edges or infinitely small receiver points.[26]

(iii) **Geometric coherence**: Tracing beams can improve the efficiency of multiple ray intersection tests. In particular, once a beam has been traced along a certain sequence of surface intersections, generating a ray path from a source to a receiver following the same sequence requires only checking the ray path for intersections with the surfaces of the sequence, and the expensive computation of casting rays through a scene can be amortized over several ray paths. Beam tracing can be used not only to enumerate potential propagation sequences, but also to identify which elements of the scene can potentially be blockers for each sequence.[47,48] This information can be used to generate and check occlusion of sampled propagation paths quickly—i.e., in time proportional to the length of the sequence rather than the complexity of the scene.

(iv) **Progressive refinement**: Characteristics of the sound waves represented by beams can be used to guide priority-driven strategies.[9,49] For instance, estimates of the acoustic energy carried by different beams can be used to order beam tracing steps and to detect early termination criteria. This method is far more practical than precomputing a global visibility structure, such as the visibility skeleton,[50] which requires large amounts of compute time and storage, mostly for pairs of surfaces for which transport is insignificant. Instead, the proposed approach traces beams in priority order, finding propagation paths only as necessary for the required accuracy of the solution.

The main challenge of beam tracing is to develop methods that trace beams through 3D models robustly and efficiently and that generate propagation paths quickly. Although several data structures have been proposed to accelerate beam tracing computations, including ones based on binary space partitions,[27] cell adjacency graphs,[37,41,8,38,39] and layers of 2D triangulations,[40] no previous method models edge diffraction without sampling artifacts, and none provides interactive path updates in large 3D environments.

The key idea behind our method is to precompute and store spatial data structures that encode all possible sequences of surface and edges scattering of sound emanating from each audio source and then use these data structures to compute propagation paths to arbitrary observer viewpoints for real-time auralization during an interactive user session. Specifically, we use a precomputed polyhedral cell complex to accelerate beam tracing and a precomputed beam tree data structure to accelerate generation of propagation paths. The net result is that our method (1) scales to support large architectural environments, (2) models propagation due to edge diffraction, (3) finds all propagation paths up to a given termination criterion without exhaustive search or risk of undersampling, and (4) updates propagation paths at interactive rates. We use this system for interactive acoustic design of architectural environments.

IV. IMPLEMENTATION

Execution of our system proceeds in four distinct phases, as shown in Fig. 6. The first two phases execute

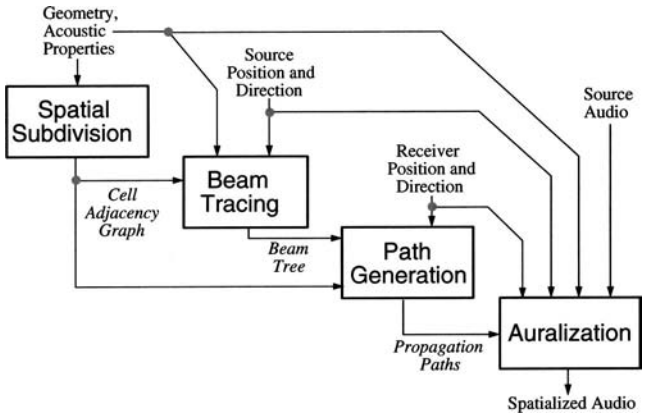


FIG. 6. System organization.

off-line, precomputing data structures for each stationary audio source, while the last two execute in real-time as a user moves the audio receiver interactively.

The result of each phase is shown in Fig. 7. First, during the *spatial subdivision phase*, we precompute spatial relationships inherent in the set of polygons describing the environment and represent them in a cell adjacency graph data structure that supports efficient traversals of space [Fig. 7(a)]. Second, during the *beam tracing phase*, we recursively follow beams of transmission, diffraction, and specular reflection through space for each audio source [Fig. 7(b)]. The output of the beam tracing phase is a beam tree data structure that explicitly encodes the region of space reachable by each sequence of reflection and transmission paths from each source point. Third, during the *path generation* phase, we compute propagation paths from each source to the receiver via lookup into the precomputed beam tree data structure as the receiver is moved under interactive user control [Fig. 7(c)]. Finally, during the *auralization phase*, we output a spatialized audio signal by convolving anechoic source signals with impulse response filters derived from the lengths, attenuations, and directions of the computed propagation paths [Fig. 7(d)]. The spatialized audio output is synchronized with real-time graphics output to provide an interactive audio/visual experience. The following subsections describe each of the four phases in detail.

A. Spatial subdivision

During the first preprocessing phase, we build a spatial subdivision representing a decomposition of 3D space and store it in a structure which we call a *winged-pair* representation. The goal of this phase is to partition space into convex polyhedral cells whose boundaries are aligned with polygons of the 3D input model and to encode cell adjacencies in a data structure enabling efficient traversals of 3D space during the later beam tracing phase.

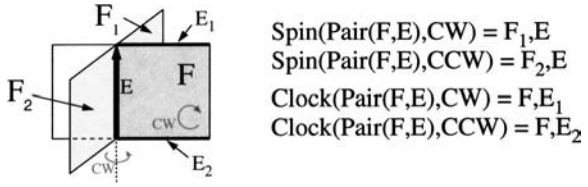


FIG. 8. Winged-pair structure.

The winged-pair data structure stores topological adjacencies in fixed-size records associated with vertices, edges, faces, cells, and face-edge pairs. Specifically, every vertex stores its 3D location and a reference to any one attached edge; every edge stores references to its two vertices and any one attached face-edge pair; every face stores references to its two cells and one attached face-edge pair; and every cell stores a reference to any one attached face. Each face-edge pair stores references to one edge *E* and one face *F* adjacent to one another, along with a fixed number of adjacency relationships useful for topological traversals. Specifically, they store references (*spin*) to the two face-edge pairs reached by spinning *F* around *E* clockwise and counter-clockwise (see Fig. 8) and to the two face-edge pairs (*clock*) reached by moving around *F* in clockwise and counter-clockwise directions from *E* (see Fig. 8). The face-edge pair also stores a bit (*direction*) indicating whether the orientation of the vertices on the edge is clockwise or counter-clockwise with respect to the face within the pair. These simple, fixed-size structures make it possible to execute efficient topological traversals of space through cell, face, edge, and vertex adjacency relationships in a manner similar to the winged-edge[51] and facet-edge structures.[52]

We build the winged-pair data structure for a 3D polygonal model using a binary space partition (BSP),[53] a recursive binary split of 3D space into convex polyhedral regions (*cells*) separated by planes. To construct the BSP, we recursively split cells by the planes of the input polygons using the method described in Ref. 54. We start with a single BSP cell containing the entire 3D space and consider polygons of the input 3D model one-by-one. For each BSP cell split by a polygon *P*, the corresponding winged-pair cell is split along the plane supporting *P*, and the faces and edges on the

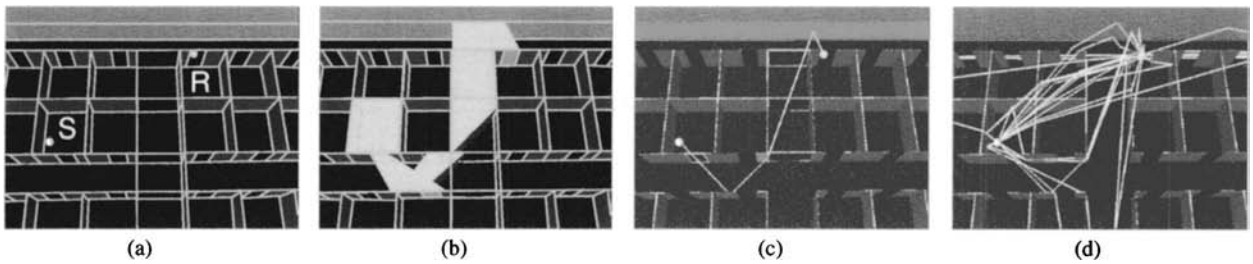


FIG. 7. Results of each phase of execution: (a) virtual environment (office cubicles) with source *S*, receiver *R*, and spatial subdivision marked in pink, (b) example reflected and diffracted beam (cyan) containing the receiver, (c) path generated for the corresponding sequence of opaque faces (green), transparent faces (purple), and edges (magenta), and (d) many paths found for different sequences from *S* to *R*.

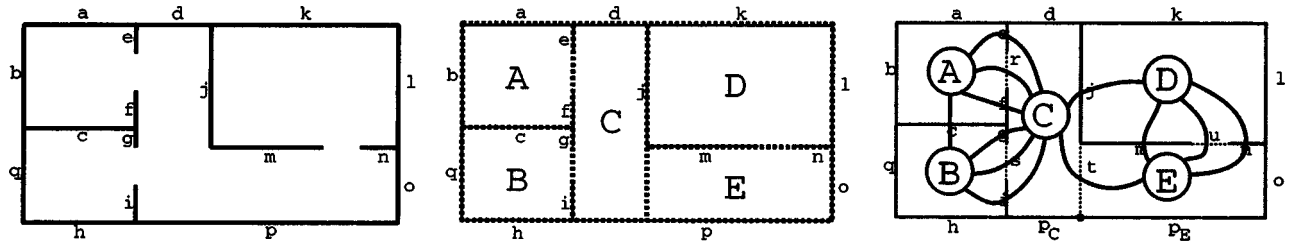


FIG. 9. Example spatial subdivision.

boundary of the cell are updated to maintain a three-manifold in which every face is flat, convex, and entirely inside or outside every input polygon. As faces are created, they are labeled according to whether they are reflectant (coincide with an input polygon) or transparent (split free space). The binary splitting process continues until no input polygon intersects the interior of any BSP cell, leading to a set of convex polyhedral cells whose faces are all convex and collectively contain all the input polygons. The resulting winged-pair is written to a file for use by later phases of our acoustic modeling process.

As an example, a simple 2D model (a) and its corresponding binary space partition (b) and cell adjacency graph (c) are shown in Fig. 9. Input ''polygons'' appear as solid line segments labeled with lower-case letters $(a-q)$; transparent cell boundaries introduced by the BSP are shown as dashed line segments labeled with lower-case letters $(r-u)$; constructed cell regions are labeled with upper-case letters $(A-E)$; and the cell adjacency graph implicit in the winged-pair structure is overlaid in Fig. 9(c).

B. Beam tracing

After the spatial subdivision has been constructed, we use it to accelerate traversals of space during beam tracing. The goal of this phase is to compute polyhedral beams representing the regions of space reachable from each stationary source by different sequences of reflections, transmissions, and diffractions. The beams are queried later during an interactive phase to compute propagation paths to specific receiver locations.

Briefly, beams are traced from each stationary sound source via a best-first traversal of the cell adjacency graph starting in the cell containing the source. As the algorithm traverses a cell boundary into a new cell, a copy of the current convex pyramidal beam is ''clipped'' to include only the region of space passing through the convex polygonal boundary to model transmissions. At each reflecting cell boundary, a copy of the transmission beam is mirrored across the plane supporting the cell boundary to model specular reflections. At each diffracting edge, a new beam is spawned whose source is the edge and whose extent includes all rays predicted by the geometric theory of diffraction.[55] The traversal along any sequence terminates when either the length of the shortest path within the beam or the cumulative attenuation exceed some user-specified thresholds. The traversal may also be terminated when the total number of beams traced or the elapsed time exceed other thresholds.

Pseudocode for the beam tracing algorithm appears in Fig. 10. Throughout the execution, a priority queue stores the set of beams to be traced sorted according to a *priority function*. Initially, the priority queue contains only one beam representing the entire space inside the cell containing the source. During each step of the algorithm, the highest priority beam B traversing a cell C is removed from the priority queue, and new ''child'' beams are placed onto the priority queue according to the following criteria:

```
void TraceBeams()
begin
    // Initialization
    S = Source point;
    D = Spatial subdivision;
    B = Beam containing all of space;
    C = Current cell;
    Q = Queue of beam tree nodes;

    C = FindCell(D, S);
    N = CreateNode(NULL, B, C);
    Q = InitQueue();

    PushQueue(Q, N);
    while (N = PopQueue(Q)) do
        // Consider each polygon on cell boundary
        foreach polygon P on boundary of N.C do
            // Check if polygon intersects beam
            if (Intersects(P, N.B)) then
                // Compute intersection beam
                Bt = Intersection(B, Beam(S, P));

                // Iterate along transmission paths
                if (Transmissive(P)) then
                    Ct = NeighborCell(D, C, P);
                    PushQueue(Q, CreateNode(N, Bt, Ct));
                endif

                // Iterate along reflection paths
                if (Reflective(P)) then
                    Br = Mirror(Bt, P);
                    PushQueue(Q, CreateNode(N, Br, C));
                endif

                // Iterate along diffraction paths
                foreach edge E on boundary of P do
                    // Check if edge intersects beam
                    if (Intersects(E, N.B)) then
                        Bd = CreateBeam(E);
                        PushQueue(Q, CreateNode(N, Bd, C));
                    endif
                endfor
            endif
        endfor
    endwhile
end
```

FIG. 10. Pseudocode for the beam tracing algorithm.

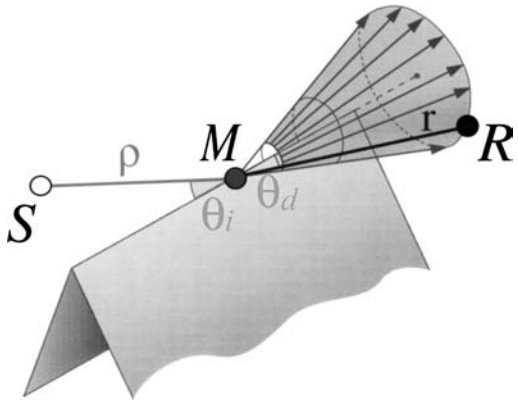


FIG. 11. Rays diffracted by a 3D edge according to the uniform theory of diffraction. The angle θ_d of the cone is equal to the angle θ_i between the incident ray and the edge.

(i) **Transmission beams**: For each nonopaque face F on the boundary of cell C and intersected by the beam B, a pyramidal *transmission beam* B_t is constructed whose apex is the source of B and whose sides each contain an edge of $F \cap B$. This new beam B_t represents sound rays traveling along B that are transmitted through F into the cell C_t which is adjacent to C across F.

(ii) **Specular reflection beams**: For each reflective face F on the boundary of cell C and intersected by the beam B, a polyhedral *specular reflection beam* B_r is constructed whose apex is the virtual source of B, created by mirroring the source of B over the plane containing F, and whose sides each contain an edge of $F \cap B$. This new beam B_r represents sound rays traveling along B that reflect specularly off of F and back into cell C.

(iii) **Diffraction beams**: For each edge E shared by two scattering faces F_1 and F_2 on the boundary of cell C and intersected by beam B, a *diffraction beam* is formed whose source is the line segment describing E and whose polyhedral extent contains the cone of potential diffraction paths bounded by the solid wedge of opaque surfaces sharing E, as shown in Fig. 11. This conservatively approximate beam contains all potential paths of sound initially traveling along B and then diffracted by edge E. For efficiency, the user may specify that diffraction beams should be traced only into shadow regions, in which case an extra half-space representing the shadow boundary is added to the beam.

Figure 12 contains an illustration of the beam tracing algorithm execution for the simple 2D example model shown in Fig. 9. The best-first traversal starts in the cell (labeled "D") containing the source point (labeled "S") with a beam containing the entire cell (D). Beams are created and traced for each of the six boundary polygons of cell "D" (j, k, l, m, n, and u). For example, transmission through the cell boundary labeled "u" results in a beam (labeled T_u) that is trimmed as it enters cell "E." T_u intersects only the polygon labeled "o," which spawns a reflection beam (labeled $T_u R_o$).

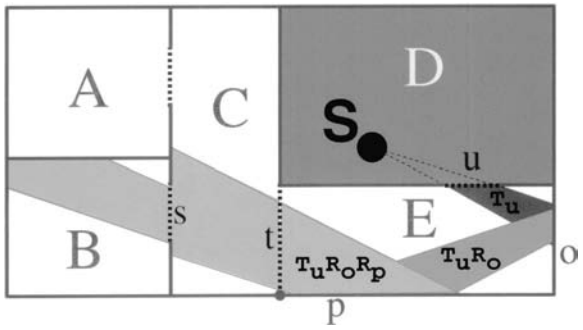


FIG. 12. Beam tracing through cell adjacency graph (this figure shows only one beam, while many such beams are traced along different propagation sequences).

That beam intersects only the polygon labeled "p," which spawns a reflection beam (labeled $T_u R_o R_p$), and so on.

Figure 13 shows an example in 3D with one sequence of beams traced up to one reflection from a source (on left) through the spatial subdivision (thin lines are cell boundaries) for a simple set of input polygons.

If the source is not a point, but instead distributed in a region of space (e.g., for diffracting edges), the exact region of space reachable by rays transmitted or reflected by a sequence of convex polygons can become quite complex, bounded by quadric surfaces corresponding to triple-edge (EEE) events.[56] Rather than representing these complex regions exactly, we conservatively overestimate the potential space of paths from each region of space edge with a convex polyhedron bounded by a fixed number of planes (usually six, as in Ref. 39). We correct for this approximation later during path generation by checking each propagation path to determine if it lies in the overestimating part of the polyhedron, in which case it is discarded. Since propagation patterns can be approximated conservatively and tightly with simple convex polyhedra, and since checking propagation paths is quick, the whole process is much more robust and faster than computing the exact propagation pattern directly. Using the adjacency information in the winged-pair structure, each new beam is constructed in constant time.

The results of the beam tracing algorithm are stored in a *beam tree* data structure[28] to be used later during path generation for rapid determination of propagation paths from the

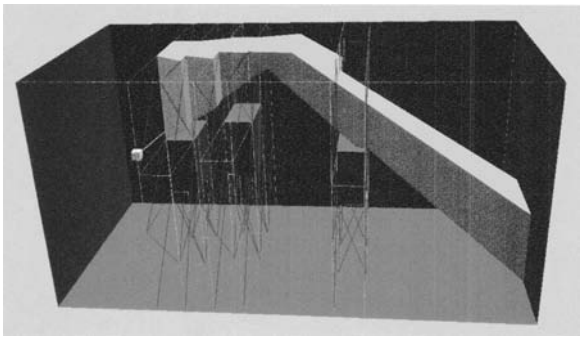


FIG. 13. A beam clipped and reflected at cell boundaries (this figure shows only one beam, while many such beams are traced along different propagation sequences).

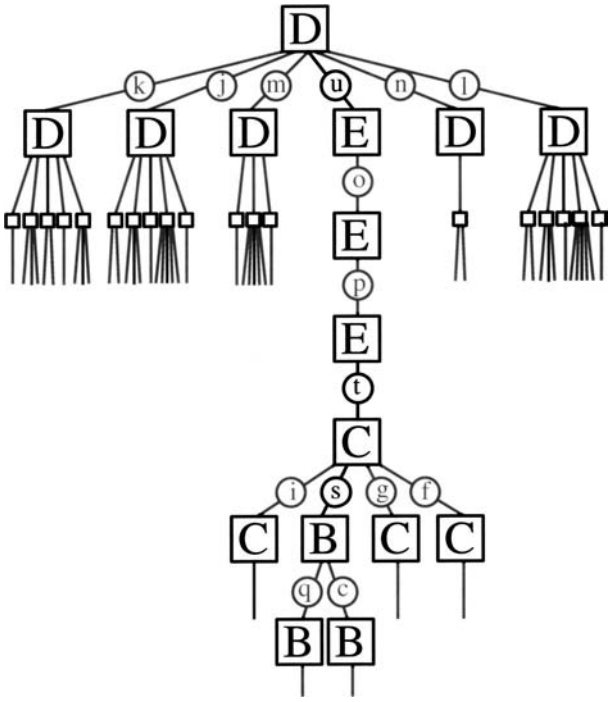


FIG. 14. Beam tree.

source point. The beam tree contains a node for each beam considered during the beam tracing algorithm. Specifically, each node stores (1) a reference to the cell being traversed, (2) a reference to the edge/face most recently traversed (if there is one), and (3) the convex polyhedral beam representing the region of space potentially reachable by the traversed sequence of transmissions, reflections, and diffractions. To further accelerate evaluation of propagation paths during a later interactive phase, each node of the beam tree also stores the cumulative attenuation due to reflective and transmissive absorption, and each cell of the spatial subdivision stores a list of ''back-pointers'' to its beam tree nodes. Figure 14 shows a partial beam tree corresponding to the traversal shown in Fig. 12.

C. Path generation

In the third phase, as a user moves the receiver interactively through the environment, we use the precomputed beam trees to identify propagation sequences of transmissions, reflections, and diffractions potentially reaching the receiver location.

Since every beam contains all points potentially reachable by rays traveling along a particular propagation sequence, we can quickly enumerate the potential propagation sequences by finding all the beams containing the receiver location. Specifically, we first find the cell containing the receiver by a logarithmic-time search of the BSP. Then, we check each beam tree node, T, associated with that cell to see whether the beam stored with T contains the receiver. If it does, a potential propagation sequence from the source point to the receiver point has been found, and the ancestors of T in the beam tree explicitly encode the set of reflections, diffractions, and transmissions through the boundaries of the

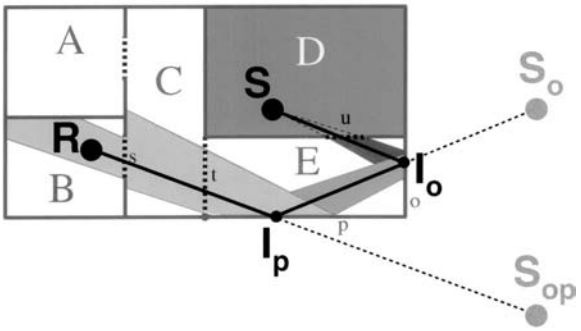


FIG. 15. Propagation path to receiver point (''R'') for the example in Figs. 12 and 14 computed via lookup in beam tree for source point (''S'').

spatial subdivision that a ray must traverse from the source to the receiver along this sequence (more generally, to any point inside the beam stored with T).

For each such propagation sequence, we construct explicit propagation path(s) from the source to the receiver. In our current system, we compute a single propagation path for each sequence as the one that is the shortest among all possible piecewise-linear paths from the source to the receiver (this path is used directly for modeling transmission, specular reflection, and diffraction according to the geometrical theory of diffraction). In order to construct this shortest path, we must find the points of intersection of the path with every face and edge in the sequence.

For sequences containing only transmissions and specular reflections (i.e., no edge diffractions), the shortest propagation path is generated analytically by iterative intersection with each reflecting surface. For instance, to find a path between a specific pair of points, S and R, along a sequence of specularly reflecting polygons P_i for $i=1,...,n$, we first traverse the polygon sequence in forward order to construct a stack of mirror images of S, where S_i corresponds to the image resulting from mirroring S over the first i of the n reflecting polygons in the sequence. Then, we construct the propagation path by traversing the polygon sequence in backward order, computing the ith vertex, V_i, of the path as the intersection of the line between V_{i-1} and S_{n-i+1} with the surface of polygon P_{n-i+1}, where V_0 is the receiver point. If every vertex V_i of the path lies within the boundary of the corresponding polygon P_i, we have found a *valid* reflection path from S to R along P. Otherwise, the path is in an overestimating part of the beam, and it can be ignored. Figure 15 shows the valid specular reflection path from the source (labeled ''S'') to a receiver (labeled ''R'') for the example shown in Fig. 12.

For sequences also containing diffracting edges, construction of the shortest propagation path is more difficult since it requires determining the locations of ''diffraction points,'' $D_i\,(i=1,...,n)$, for the n diffracting edges. These diffraction points generally lie in the interior of the diffracting edges (see Fig. 16), and the path through them locally satisfies a simple ''unfolding property:'' the angle (θ_i) at which the path enters each diffracting edge must be the same as the angle (ϕ_i) at which it leaves.[57] (The unfolding property is a consequence of the generalized Fermats principle.[55]) Thus, to find the shortest path through n diffracting edges

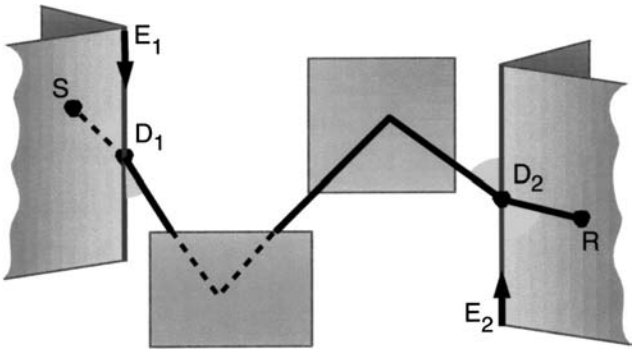


FIG. 16. A single propagation path comprising a diffraction, two specular reflections, and another diffraction. The two diffraction points (D_i) are determined by equal angle constraints at the corresponding edges (E_i).

and m transmitting and/or specularly reflecting faces, we must solve a nonlinear system of n equations expressing equal angle constraints at all diffracting edges:

$$\begin{aligned}
\overrightarrow{D_1S}\cdot\vec{E_1} &= \overrightarrow{D_1D_2}\cdot(-\vec{E_1})\\
\overrightarrow{D_2D_1}\cdot\vec{E_2} &= \overrightarrow{D_2D_3}\cdot(-\vec{E_2})\\
&\vdots\\
\overrightarrow{D_nD_{n-1}}\cdot\vec{E_n} &= \overrightarrow{D_nR}\cdot(-\vec{E_n})
\end{aligned} \tag{1}$$

where S is the source position, R is the receiver position, $\vec{E_i}$ is the normalized direction vector of the ith diffracting edge, and $\overrightarrow{D_{i+1}D_i}$ is a normalized direction vector between two adjacent points in the shortest path. To incorporate specular reflections in this equation, $\vec{E_i}$ and $\overrightarrow{D_{i+1}D_i}$ are both transformed by a mirroring operator accounting for the sequence of specularly reflecting faces up to the ith diffraction.

Parametrizing the edges, $D_i = O_i + t_i\vec{E_i}$ (where O_i is a reference point on edge i), the system of equations (1) can be rewritten in terms of n unknowns (t_i) and solved within a specified tolerance using a nonlinear system solving scheme. We use a locally convergent Newton scheme,[58] with the middle of the edges as a starting guess for the diffraction points. Since the equation satisfied by any diffraction point only depends on the previous and next diffraction points in the sequence, the Jacobian matrix is tridiagonal and can easily be evaluated analytically. Thus, every Newton iteration can be performed in time $O(n)$ where n is the number of unknowns (i.e., edges). We found this method to be faster than the recursive geometrical construction proposed by Aveneau.[59]

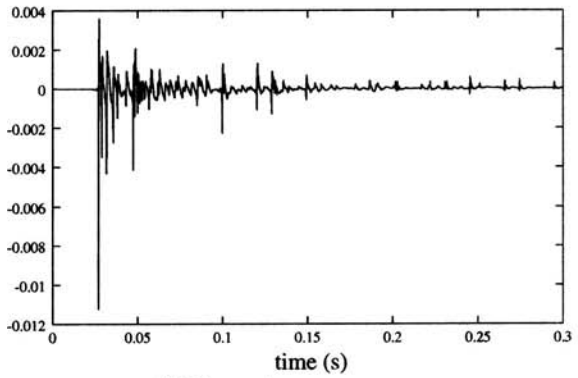
Once the intersection points of a propagation path are found, we validate whether the path intersects every surface and edge in the sequence (to compensate for the fact that the beams are conservatively approximate). If not, the path belongs to the overestimating part of the beam and is discarded. Otherwise, it contributes to an *impulse response* used for spatializing sound.

D. Auralization

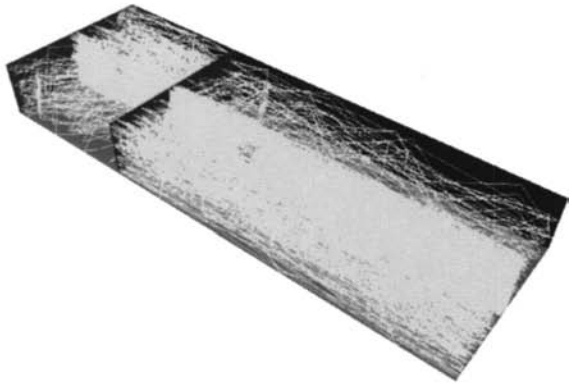
The computed early propagation paths are combined with a statistical approximation of late reverberation to model an impulse response of the virtual environment for every source/receiver pair. Dry sound signals emanating from the source are convolved with this digital filter to produce spatialized audio output.

Although this paper focuses on computation of geometric propagation paths, for the sake of completeness, we describe two auralization methods implemented in our system: (1) an off-line, high-resolution method for applications in which accuracy is favored over speed, and (2) an on-line, low-resolution approximation suitable for interactive walkthrough applications. Please refer to other papers (e.g., Refs. 7 and 60) for more details on auralization methods.

In the off-line case, we compute the early part of the impulse response in the Fourier frequency domain at the sampling rate resolution (e.g., 8000 complex values are updated for every propagation path for a one second long response at 16 kHz). As an example, Fig. 17 shows an impulse response computed for up to ten orders of specular reflections between source and receiver points in coupled-rooms. Our implementation includes frequency-dependent source and head filtering effects (obtained through measurements) and material filtering effects (derived from either measure-



(a) Impulse response



(b) Propagation paths

FIG. 17. Impulse response (left) computed for up to ten orders of specular reflections (right) between a point source (B&K "artificial mouth") and point receiver (omnidirectional) in a coupled-rooms environment (two rooms connected by an open door). There are 353 paths. The small room is $7\times8\times3$ m^3, while the large room is $17\times8\times3$ m^3.

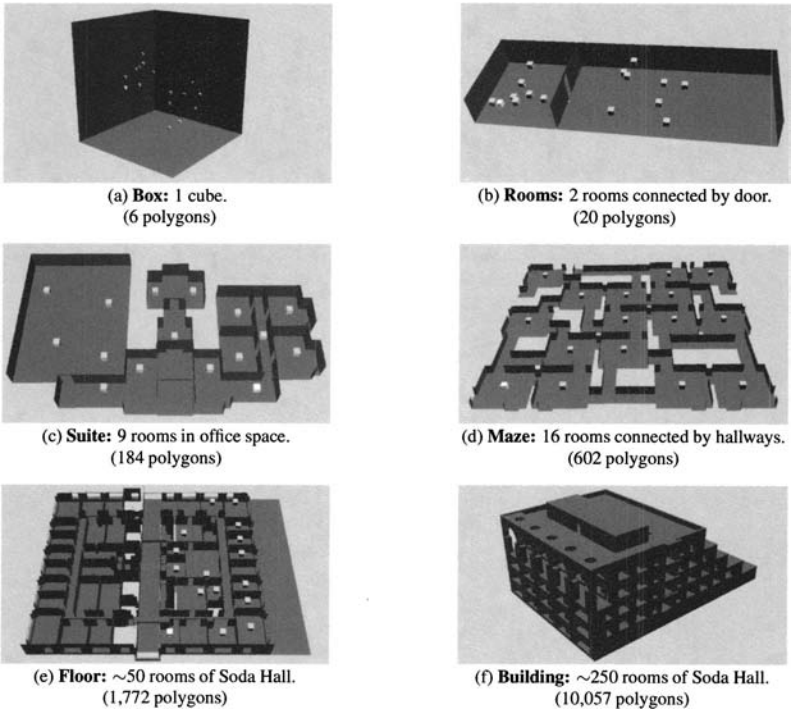


FIG. 18. Test models (source locations are gray dots).

ments or analytical models). We derive analytical models for the frequency-dependent impedance from the Delany–Bazley formula[61] and for the pressure reflection coefficient from the well-known plane wave formula of Pierce[62] (p. 33). Other wave formulas, such as the exact expression for the reflection of a spherical wave off an infinite impedant surface, derived by Thomasson,[63] could be used for improved accuracy in the near field from the surfaces.

We compute frequency-dependent diffraction coefficients using the uniform theory of diffraction[55,64,65] applied along the shortest paths constructed by our algorithm. If more accuracy is required, the information given in computed propagation sequences (exact intersected portions of surfaces and edges) can be used to derive filters, for example based on more recent results exploiting the Biot–Tolstoy–Medwin approach.[66–69] In this case, the shortest path computed by our algorithms can still be used to determine efficiently a unique incident direction on the listener's head for binaural processing (as suggested in Ref. 69).

In the on-line case, our system auralizes sound in real-time as the receiver position moves under interactive user control. A separate, concurrently executing process is spawned to perform convolution in software. To provide plausible spatialization with limited processing resources, we use a small number of frequency bands to reequalize and delay the source signal for every path, computing its contribution to a stereo impulse response in the time domain.[60] The delay associated with each path is given by L/C, where L is the length of the corresponding propagation path, and C is the speed of sound. The amplitude is given by A/L, where A is the product of all the attenuation coefficients for the reflecting, diffracting, and transmitting surfaces along the corresponding propagation sequence. Stereo impulse responses are generated by multiplying the amplitude of each path by the cardioid directivity function $((1+\cos(\theta))/2)$, where θ is the angle of arrival of the pulse with respect to the normal vector pointing out of the ear) corresponding to each ear. These gross approximations enable our auralization to give real-time feedback with purely software convolution. Other methods utilizing DSP hardware (e.g., binaural presentation) could easily be incorporated into our system in the future.

V. RESULTS

The 3D data structures and algorithms described in the preceding sections have been implemented in C++ and run on Silicon Graphics and PC/Windows computers.

To test whether the algorithms support large 3D environments and update propagation paths at interactive rates, we performed a series of experiments in which propagation paths were computed in a variety of architectural models (shown in Fig. 18). The test models ranged from a simple box with 6 polygons to a complex building with over 10 000 polygons. The experiments were run on a Silicon Graphics Octane workstation with 640 MB of memory and used one 195 MHz R10000 processor.

The focus of the experiments is to compare the computational efficiency of our method with image source methods, the approach most commonly used for interactive acoustic modeling applications. Accordingly, for the sake of direct comparison, we limited our beam tracing system to consider only specular reflections. In this case, our beam tracing method produces exactly the same set of propagation paths as classical image source methods. However, as we shall see, our beam tracing method has the ability to scale to large environments and to generate propagation paths at interactive rates.

In each experiment, we measured the time and storage required for spatial subdivision, beam tracing, sequence con-

TABLE I. Spatial subdivision statistics.

Model name	Input polys	Cell regions	Cell boundaries	Time (s)	Storage (MB)
Box	6	7	18	0.0	0.004
Rooms	20	12	43	0.1	0.029
Suite	184	98	581	3.0	0.352
Maze	602	172	1187	4.9	0.803
Floor	1772	814	5533	22.7	3.310
Bldg	10 057	4512	31 681	186.3	18.694

struction, and path generation. Results are reported in the following subsections.

A. Spatial subdivision results

We first constructed the spatial subdivision data structure (cell adjacency graph) for each test model. Statistics from this phase of the experiment are shown in Table I. Column 2 lists the number of input polygons in each model, while columns 3 and 4 contain the number of cell regions and boundary polygons, respectively, generated by the spatial subdivision algorithm. Column 5 contains the wall-clock execution time (in seconds) for the algorithm, while column 6 shows the storage requirements (in MBs) for the resulting spatial subdivision.

Empirically, we find that the number of cell regions and boundary polygons grows linearly with the number of input polygons for typical architectural models (see Fig. 19), rather than quadratically as is possible for worst case geometric arrangements. The reason for linear growth is illustrated in the two images inlaid in Fig. 19, which compare spatial subdivisions for the Maze test model (on the left) and a 2×2 grid of Maze test models (on the right). The 2×2 grid of Mazes has exactly four times as many polygons and approximately four times as many cells. The storage requirements of the spatial subdivision data structure also grow linearly as they are dominated by the vertices of boundary polygons.

The time required to construct the spatial subdivisions grows super-linearly, dominated by the code that selects and orders splitting planes during BSP construction (see Ref. 54). However, it is important to note that the spatial subdivision phase need be executed only once off-line for each geometric model, as its results are stored in a file, allowing rapid reconstruction in subsequent beam tracing executions.

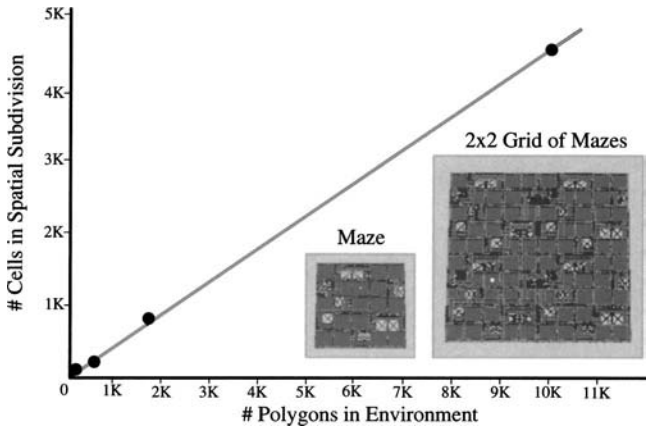


FIG. 19. Plot of subdivision size versus polygonal complexity.

TABLE II. Beam tracing and path generation statistics.

Model name	No. of polys	No. of Rfl	Beam tracing		Path generation	
			No. of beams	Time (ms)	No. of paths	Time (ms)
Box	6	0	1	0	1.0	0.0
		1	7	1	7.0	0.1
		2	37	3	25.0	0.3
		4	473	42	129.0	6.0
		8	10 036	825	833.0	228.2
Rooms	20	0	3	0	1.0	0.0
		1	31	3	7.0	0.1
		2	177	16	25.1	0.3
		4	1939	178	127.9	5.2
		8	33 877	3024	794.4	180.3
Suite	184	0	7	1	1.0	0.0
		1	90	9	6.8	0.1
		2	576	59	25.3	0.4
		4	7217	722	120.2	6.5
		8	132 920	13 070	672.5	188.9
Maze	602	0	11	1	0.4	0.0
		1	167	16	2.3	0.0
		2	1162	107	8.6	0.1
		4	13 874	1272	36.2	2.0
		8	236 891	21 519	183.1	46.7
Floor	1772	0	23	4	1.0	0.0
		1	289	39	6.1	0.1
		2	1713	213	21.5	0.4
		4	18 239	2097	93.7	5.3
		8	294 635	32 061	467.0	124.5
Bldg	10 057	0	28	5	1.0	0.0
		1	347	49	6.3	0.1
		2	2135	293	22.7	0.4
		4	23 264	2830	101.8	6.8
		8	411 640	48 650	529.8	169.5

B. Beam tracing results

We tested our beam tracing algorithm with 16 source locations in each test model. The source locations were chosen to represent typical audio source positions (e.g., in offices, in common areas, etc.)—they are shown as gray dots in Fig. 18 (we use the same source locations in Building model as in the Floor model). For each source location, we traced beams (i.e., constructed a beam tree) five times, each time with a different limit on the maximum order of specular reflections (e.g., up to 0, 1, 2, 4, or 8 orders). Other termination criteria based on attenuation or path length were disabled, and transmission was ignored, in order to isolate the impact of input model size and maximum order of specular reflections on computational complexity.

Table II contains statistics from the beam tracing experiment—each row represents a test with a particular 3D model and maximum order of reflections, averaged over all 16 source locations. Columns 2 and 3 show the number of polygons describing each test model and the maximum order of specular reflections allowed in each test, respectively. Column 4 contains the average number of beams traced by our

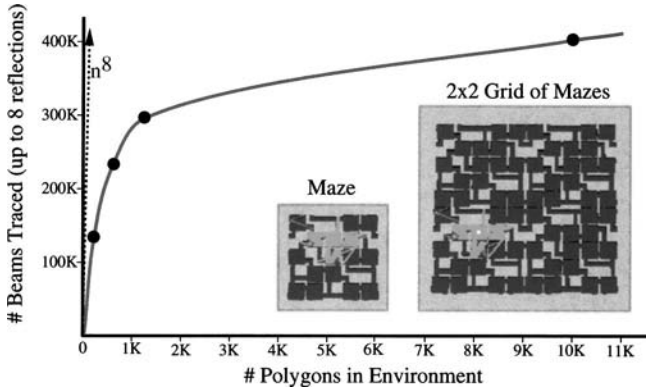


FIG. 20. Plot of beam tree size versus polygonal complexity.

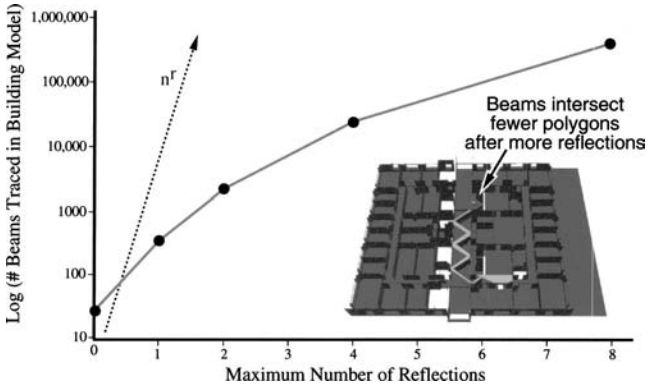


FIG. 21. Plot of beam tree size with increasing reflection orders.

algorithm (i.e., the average number of nodes in the resulting beam trees), and column 5 shows the average wall-clock time (in milliseconds) for the beam tracing algorithm to execute.

1. Scale with increasing polygonal complexity

We readily see from the results in column 4 that the number of beams traced by our algorithm (i.e., the number of nodes in the beam tree) does *not* grow at an exponential rate with the number of polygons (n) in these environments (as it does using the image source method). Each beam traced by our algorithm preclassifies the regions of space according to whether the corresponding virtual source (i.e., the apex of the beam) is visible to a receiver. Rather than generating a virtual source (beam) for every front-facing surface at each step of the recursion as in the image source method, we directly find only the potentially visible virtual sources via beam-polygon intersection and cell adjacency graph traversal. We use the current beam and the current cell of the spatial subdivision to find the small set of polygon reflections that admit visible higher-order virtual sources.

The benefit of this approach is particularly important for large environments in which the boundary of each convex cell is simple, and yet the entire environment is very complex. As an example, consider computation of up to eighth order specular reflections in the Building test model (the last row of Table II). The image source method must consider approximately 1 851 082 741 virtual sources ($\Sigma_{r=0}^{8}(10\,057/2)^r$), assuming half of the 10 057 polygons are front-facing to each virtual source. Our beam tracing method considers only 411 640 virtual sources, a difference of four orders of magnitude. In most cases, it would be impractical to build and store the recursion tree without such effective pruning.

In ''densely occluded'' environments, in which all but a little part of the environment is occluded from any source point (e.g., most buildings and cities), the number of beams traced by our algorithm even grows sublinearly with the total number of polygons in the environment (see Fig. 20). In these environments, the number of sides to each polyhedral cell is nearly constant, and a nearly constant number of cells are reached by each beam, leading to near-constant expected-case complexity of our beam tracing algorithm with increasing global environment complexity.

This result is most readily understood by comparing the number of beams traced for up to eighth order reflections in the Floor and Building models (i.e., the rightmost two data points in Fig. 20). The Floor model represents the fifth floor of Soda Hall at UC Berkeley. It is a subset of the Building model, which represents five floors of the same building (floors 3–7 and the roof). Although the Building model (10 057 polygons) has more than five times the complexity of the Floor model (1772 polygons), the average number of beams traced from the same source locations by our algorithm is only 1.2–1.4 times larger for the Building model (e.g., 411 640/294 635=1.4). This is because the complexity of the spatial subdivision on the fifth floor of the building is similar in both cases, and most other parts of the building are not reached by any beam. Similarly, we expect that the beam tracing algorithm would have nearly the same complexity if the entire building were 1000 floors high, or if it were in a city of 1000 buildings. This result is shown visually in Fig. 20: the number of beams (green) traced in the Maze test model (left) does not increase significantly if the model is increased to be a 2×2 grid of Maze models (right). *The beam tracing algorithm is impacted only by local complexity, and not by global complexity.*

2. Scale with increasing reflections

We see that the number of beams traced by our algorithm grows exponentially as we increase the maximum order of reflections (r), but far slower than $O(n^r)$ as in the image source method. Figure 21 shows a logscale plot of the average number of beams traced in the Building model with increasing orders of specular reflections. The beam tree growth is less than $O(n^r)$ because each beam narrows as it is clipped by the cell boundaries it has traversed, and thus it tends to intersect fewer cell boundaries (see the example beam inlaid in Fig. 21). In the limit, each beam becomes so narrow that it intersects only one or two cell boundaries, on average, leading to a beam tree with a small branching factor (rather than a branching factor of $O(n)$, as in the image source method).

As an example, consider Table III which shows the average branching factor for nodes at each depth of the beam tree constructed for up to eighth order specular reflections in the Building model from one source location. The average

TABLE III. Example beam tree branching statistics. The number of interior nodes (have children) and leaf nodes (have no children) at each depth are listed, along with the average branching factor for interior nodes.

Tree depth	Total nodes	Interior nodes	Leaf nodes	Branching factor
0	1	1	0	16.0000
1	16	16	0	6.5000
2	104	104	0	4.2981
3	447	446	1	2.9193
4	1302	1296	6	2.3920
5	3100	3092	8	2.0715
6–10	84 788	72 469	12 319	1.2920
11–15	154 790	114 664	40 126	1.2685
>15	96 434	61 079	35 355	1.1789

branching factor (column 5) generally decreases with tree depth and is generally bounded by a small constant in lower levels of the tree.

C. Path generation results

In order to verify that specular reflection paths are computed at interactive rates from stationary sources as the receiver moves, we conducted experiments to quantify the complexity of generating specular reflection paths to different receiver locations from precomputed beam trees. For each beam tree in the previous experiment, we logged statistics during generation of specular propagation paths to 16 different receiver locations. Receivers were chosen randomly within a two foot sphere around the source to represent a typical audio scenario in which the source and receiver are in close proximity within the same ''room.'' We believe this represents a worst-case scenario as fewer paths would likely reach more remote and more occluded receiver locations.

Columns 6 and 7 of Table II contain statistics gathered during path generation for each combination of model and termination criterion averaged over all 256 source-receiver pairs (i.e., 16 receivers for each of the 16 sources). Column 6 contains the average number of propagation paths generated, while column 7 shows the average wall-clock time (in milliseconds) for execution of the path generation algorithm. Figure 22 shows a plot of the wall-clock time required to generate up to eighth order specular reflection paths for each test model.

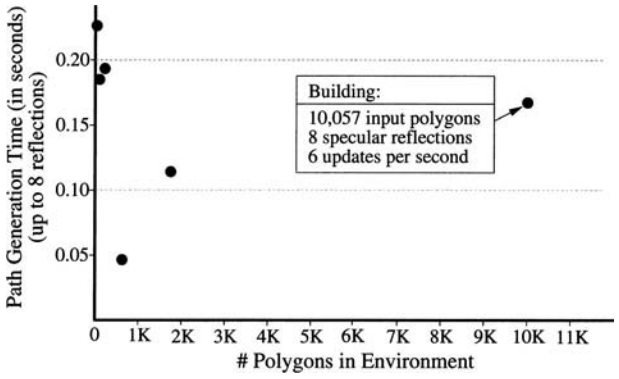


FIG. 22. Path compute time versus polygonal complexity.

We find that the number of specular reflection paths is nearly constant across all of our test models when a source and receiver are located in close proximity of one another. Also, the time required by our path generation algorithm is generally *not* dependent on the number of polygons in the environment (see Fig. 22), nor is it dependent on the total number of nodes in the precomputed beam tree. This result is due to the fact that our path generation algorithm considers only nodes of the beam tree with beams residing inside the cell containing the receiver location. Therefore, the computation time required by the algorithm is *not* dependent on the complexity of the environment outside the receiver's cell, but instead on the number of beams that traverse the receiver's cell.

Overall, we find that our algorithm supports generation of specular reflection paths between a fixed source and any (arbitrarily moving) receiver at interactive rates in complex environments. For instance, we are able to compute up to eighth order specular reflection paths in the Building environment with more than 10 000 polygons at a rate of approximately six times per second (i.e., the rightmost point in the plot of Fig. 22).

VI. DISCUSSION

In this paper, we describe beam tracing algorithms and data structures that accelerate computation of propagation paths in large architectural environments. The following subsections discuss applications of the proposed methods, limitations of our approach, and related topics for further study.

A. Applications

There are several potential applications for the methods proposed in this paper. For instance, traditional accoustical design programs (e.g., CATT Acoustics[12]) could be enhanced with real-time auralization and visualization that aid a user in understanding which surfaces cause particular acoustical effects.

Alternatively, real-time acoustic simulation can be used to enhance simulation of virtual environments in interactive walkthrough applications. Auditory cues are important in immersive applications as they can combine with visual cues to

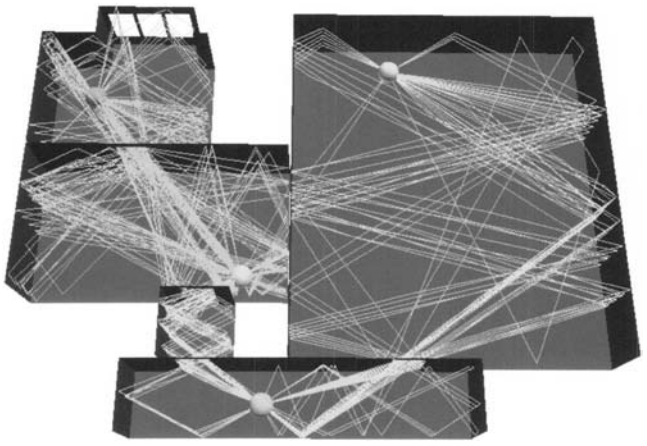


FIG. 23. Sound propagation paths (lines) between four avatars (spheres) representing users in shared virtual environment.

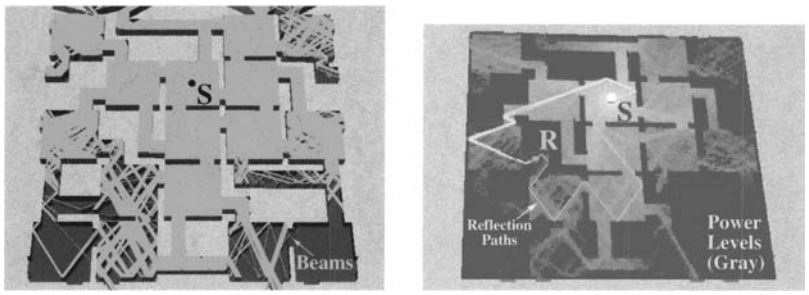


FIG. 24. Eighth-order specular reflection beams (left) and predicted power levels (right) in Maze model.

aid localization of objects, separation of simultaneous sound signals, and formation of spatial impressions of an environment.[70] For instance, binaural auditory cues are helpful in localizing objects outside a user's field of view, such as when a car comes around a blind corner in a driving simulation. They also help the separation of simultaneous sounds (e.g., many speakers at a cocktail party). Finally, qualitative changes in sound propagation, such as more absorption in a room with more plush carpets, can enhance and reinforce visual comprehension of the environment. Experiments have shown that more accurate acoustic modeling provides a user with a stronger sense of presence in a virtual environment.[2]

We have integrated our beam tracing method into an immersive system that allows a user to move through a virtual environment while images and spatialized audio are rendered in real-time according to the user's simulated viewpoint.[8–10] In the example shown in Fig. 23, multiple users represented by avatars (spheres) sharing a virtual world can speak to one another while the system spatializes their voices according to sound propagation paths (lines) through the environment.

In order to support multiple simultaneously moving sources and receivers,[9] as is required by a distributed virtual environment application with many avatars, we can no longer precompute beam trees. Instead, we must compute them in real-time as the source moves. However, we can take advantage of the fact that sounds can only be generated or heard at the positions of ''avatars'' representing the users. This simple observation enables two important enhancements to our beam tracing method. First, a bidirectional beam tracing algorithm combines beams traced from both sources and receivers to find propagation paths between them. Second, an amortized beam tracing algorithm computes beams emanating from box-shaped regions of space containing predicted avatar locations and reuses those beams multiple times to compute propagation paths as each avatar moves inside the box. We have incorporated these two enhancements into a time-critical multiprocessing system that allocates its computational resources dynamically in order to compute the highest priority propagation paths between moving avatar locations in real-time with graceful degradation and adaptive refinement. These enhancements result in two orders of magnitude of improvement in computational efficiency in the case where beams are traced for known receiver locations. See Ref. 9 for details.

Overall, we find that precomputing beams is advantageous for stationary sound sources and arbitrarily moving receivers, while computing them asynchronously on the fly is still practical for continuously moving sources and receivers.

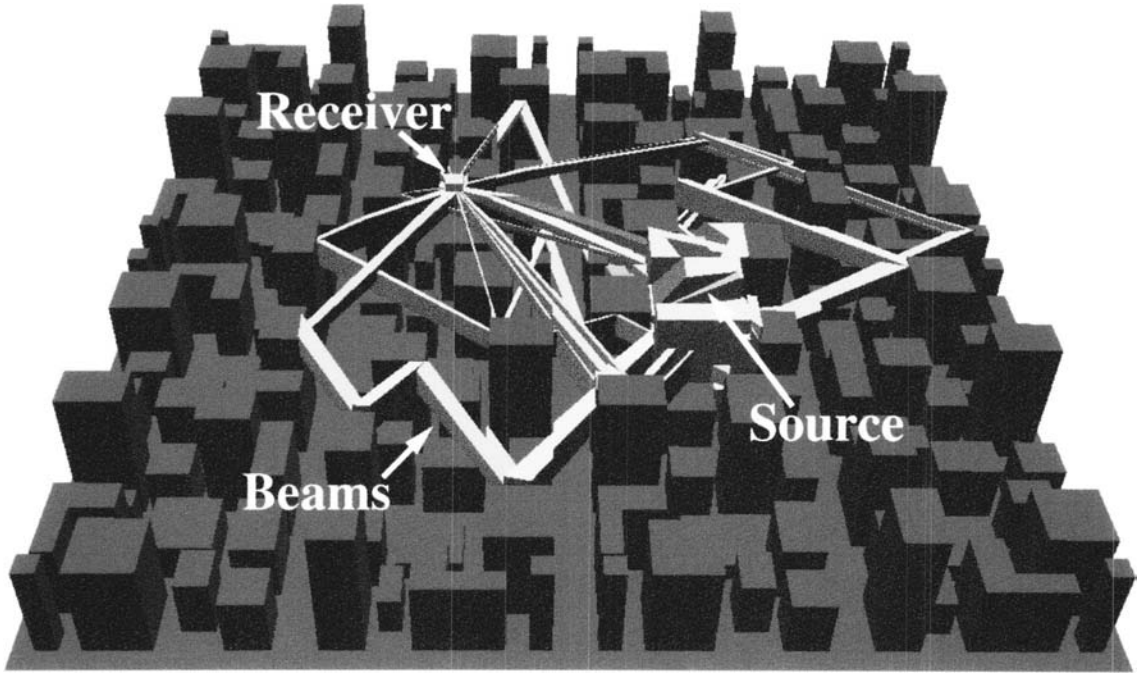


FIG. 25. Beams (green) containing all eighth-order specular reflection paths from a source to a receiver in City model.

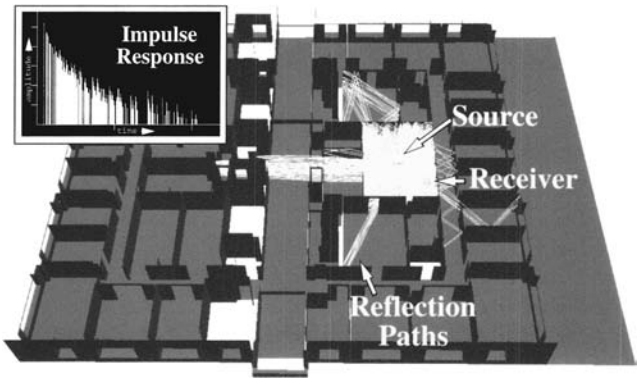


FIG. 26. Impulse response (inset) derived from eighth-order specular reflection paths (yellow) in Floor model.

B. Visualization

In order to aid the understanding and debugging of our acoustic modeling method, we find it extremely valuable to use interactive visualization of our data structures and algorithms. Our system provides menu and keyboard commands that may be used to toggle display of the (1) input polygons, (2) source point, (3) receiver point, (4) boundaries of the spatial subdivision, (5) pyramidal beams, (6) image sources, and (7) propagation paths. The system also supports visualization of acoustic metrics (e.g., power, clarity, etc.) for a set of receiver locations on a regular planar grid displayed with a textured polygon. Example visualizations are shown in Figs. 24–26.

Of course, many commercial[11–13] and research systems[29,71] provide elaborate tools for visualizing computed acoustic metrics. The critical difference in our system is that it supports continuous interactive updates of propagation paths and debugging information as a user moves the receiver point with the mouse. For instance, Figs. 24 and 26 show eighth-order specular reflection paths from a single audio source to a receiver location which is updated more than six times per second as the receiver location is moved arbitrarily. Figure 27 shows paths with specular reflections and diffractions computed in a city model and an auditorium. The user may select any propagation path for further inspection by clicking on it and then independently toggle display of reflecting cell boundaries, transmitting cell boundaries, and the polyhedral beams associated with the selected path.

Separate pop-up windows provide real-time display of other useful visual debugging and acoustic modeling information. For instance, one window shows a diagram of the beam tree data structure. Each beam tree node is dynamically colored in the diagram according to whether the receiver point is inside its associated beam or cell. Another window shows a plot of the impulse response representing the propagation paths from source to receiver (see Fig. 26). A third window shows values of various acoustic metrics, including power, clarity, reverberation time, and frequency response. All of the information displayed is updated in real-time as the user moves the receiver interactively with the mouse.

C. Geometric limitations

Our system is a research prototype, and it has several limitations. First, the 3D model must comprise only planar polygons because we do not model the transformations for beams as they reflect off curved surfaces. Furthermore, we do not trace beams along paths of refraction or diffuse reflection, which may be important acoustical effects. Each acoustic reflector is assumed to be locally reacting and to have dimensions far exceeding the wavelength of audible sound.

Second, our methods are only practical for coarse 3D models without highly faceted surfaces, such as the ones often found in acoustic modeling simulations of architectural spaces and concert halls. The difficulty is that beams are fragmented by cell boundaries as they are traced through a cell adjacency graph. For this reason, our beam tracing method would not perform well for geometric models with high local geometric complexity (e.g., a forest of trees).

Third, the major occluding and reflecting surfaces of the virtual environment must be static through the entire execution. If any acoustically significant polygon were to move, the cell adjacency graph would have to be updated incrementally.

The class of geometric models for which our method does work well includes most architectural and urban environments. In these cases, acoustically significant surfaces are generally planar, large, and stationary, and the acoustical effects of any sound source are limited to a local region of the environment (*densely occluded*).

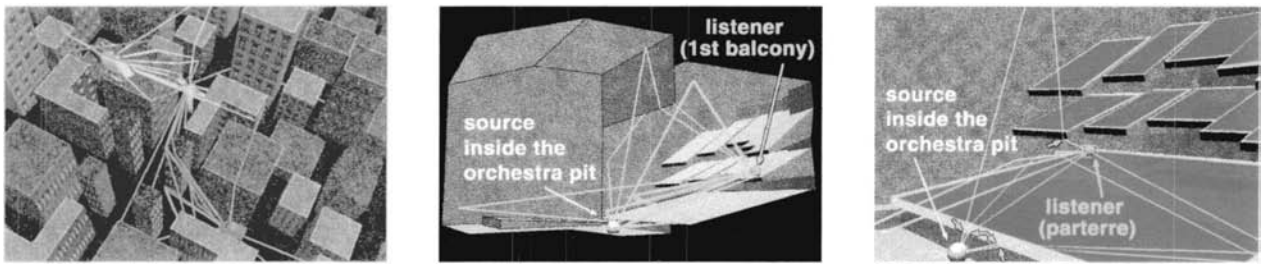


FIG. 27. Visualizations of sound paths in different environments. Diffraction of sound in a city environment is shown on the left, while early propagation paths for a source located in the orchestra pit of an opera house are shown in the middle and right images. Note the diffracted paths over the lip of the pit and balconies (cyan arrows).

D. Future work

Our system could be extended in many ways. For instance, the beam tracing algorithm is well suited for parallelization, with much of the previous work in parallel ray tracing directly applicable.[72] Also, the geometric regions covered by each node of the beam tree could be stored in a single hierarchical spatial structure (e.g., a BSP), allowing logarithmic search during path generation, rather than linear search of the beams inside a single cell. Of course, we could also use beam trees to allow a user to manipulate the acoustic properties of individual surfaces of the environment interactively with real-time feedback, such as for parametrized ray tracing[73] or for inverse modeling.[74]

Verification of our simulation results by comparison to measured data is an important topic for further study. In this paper, we have purposely focused on the computational aspects of geometrical acoustic modeling and left the validation of the models for future work. For this aspect of the problem, we refer the reader to related verification studies (e.g., Ref. 75), noting that our current system can compute the same specular reflection paths as methods based on image sources and ray tracing for which verification results are published (e.g., Refs. 76 and 77).

We are currently making impulse response measurements for verification of our simulations with reflections, diffractions, and transmissions. We have recently built a ''room'' for validation experiments. The base configuration of the room is a simple box. However, it is constructed with reconfigurable panels that can be removed or inserted to create a variety of interesting geometries, including ones with diffracting panels in the room's interior. We are currently measuring the directional reflectance distribution of each of these panels in the Anechoic Chamber at Bell Laboratories. We plan to make measurements with speakers and microphones at several locations in the room and with different geometric arrangements of panels, and we will compare the measurements with the results of simulations with the proposed beam tracing algorithms for matching configurations.

Perhaps the most interesting direction of future work is to investigate the possible applications of *interactive* acoustic modeling. What can we do with interactive manipulation of acoustic model parameters that would be difficult to do otherwise? As a first application, we hope to build a system that uses our interactive acoustic simulations to investigate the psychoacoustic effects of varying different acoustic modeling parameters. Our system will allow a user to interactively change various acoustic parameters with real-time auralization and visualization feedback. With this interactive simulation system, it may be possible to address psychoacoustic questions, such as ''how many reflections are psychoacoustically important to model?'' or ''which surface reflection model provides a psychoacoustically better approximation?'' Moreover, we hope to investigate the interaction of visual and aural cues on spatial perception. We believe that the answers to such questions are of critical importance to future design of 3D simulation systems.

VII. CONCLUSION

We have described a system that uses beam tracing data structures and algorithms to compute early propagation paths from static sources to a moving receiver at interactive rates for real-time auralization in large architectural environments.

As compared to previous acoustic modeling approaches, our beam tracing method takes unique advantage of *precomputation* and *convexity*. Precomputation is used twice, once to encode in the spatial subdivision data structure a depth-ordered sequence of (cell boundary) polygons to be considered during any traversal of space, and once to encode in the beam tree data structure the region of space reachable from a static source by sequences of specular reflections, diffractions, and transmissions at cell boundaries. We use the convexity of the beams, cell regions, and cell boundary polygons to enable efficient and robust computation of beam-polygon and beam-receiver intersections. As a result, our method is uniquely able to (1) enumerate all propagation paths robustly, (2) scale to compute propagation paths in large, densely occluded environments, (3) model effects of edge diffraction in arbitrary polyhedral environments, and (4) support evaluation of propagation paths at interactive rates. Our interactive system integrates real-time auralization with visualization of large virtual environments.

Based on our initial experiences with this system, we believe that interactive geometric acoustic modeling provides a valuable new tool for understanding sound propagation in complex 3D environments. We are continuing this research in order to further investigate the perceptual interaction of visual and acoustical effects and to better realize the opportunities possible with interactive acoustic modeling.

ACKNOWLEDGMENTS

The authors thank Sid Ahuja for his support of this work, and Arun C. Surendran and Michael Gatlin for their valuable discussions and contributions to the project.

[1] D. R. Begault, *3D Sound for Virtual Reality and Multimedia* (Academic, New York, 1994).
[2] N. I. Durlach and A. S. Mavor, *Virtual Reality Scientific and Technological Challenges*, National Research Council Report (National Academy, Washington, D.C., 1995).
[3] H. Kuttruff, *Room Acoustics (3rd edition)* (Elsevier Applied Science, New York, 1991).
[4] J. B. Allen and D. A. Berkley, ''Image method for efficiently simulating small room acoustics,'' J. Acoust. Soc. Am. **65**, 943–950 (1979).
[5] J. Borish, ''Extension of the image model to arbitrary polyhedra,'' J. Acoust. Soc. Am. **75**, 1827–1836 (1984).
[6] U. R. Krockstadt, ''Calculating the acoustical room response by the use of a ray tracing technique,'' J. Sound Vib. **8**(18), 118–125 (1968).
[7] M. Kleiner, B. I. Dalenback, and P. Svensson, ''Auralization—An overview,'' J. Audio Eng. Soc. **41**(11), 861–875 (1993).
[8] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West, ''A beam tracing approach to acoustic modeling for interactive virtual environments,'' *ACM Computer Graphics, SIGGRAPH'98 Proceedings*, July 1998, pp. 21–32.
[9] T. Funkhouser, P. Min, and I. Carlbom, ''Real-time acoustic modeling for distributed virtual environments,'' *ACM Computer Graphics, SIGGRAPH'99 Proceedings*, August 1999, pp. 365–374.
[10] N. Tsingos, T. Funkhouser, A. Ngan, and I. Carlbom, ''Modeling acoustics in virtual environments using the uniform theory of diffraction,'' *ACM Computer Graphics, SIGGRAPH 2001 Proceedings*, August 2001, pp. 545–552.

[11] Bose Corporation, Bose Modeler, Framingham, MA, http://www.bose.com.
[12] CATT-Acoustic, Gothenburg, Sweden, http://www.netg.se/catt.
[13] J. M. Naylor, "Odeon—Another hybrid room acoustical model," Appl. Acoust. **38**(1), 131–143 (1993).
[14] R. D. Ciskowski and C. A. Brebbia (eds.), *Boundary Element Methods in Acoustics* (Elseiver Applied Science, New York, 1991).
[15] P. Filippi, D. Habault, J. P. Lefevre, and A. Bergassoli, *Acoustics, Basic Physics, Theory and Methods* (Academic, New York, 1999).
[16] A. Kludszuweit, "Time iterative boundary element method (TIBEM)—a new numerical method of four-dimensional system analysis for the calculation of the spatial impulse response," Acustica **75**, 17–27 (1991) (in German).
[17] S. Kopuz and N. Lalor, "Analysis of interior acoustic fields using the finite element method and the boundary element method," Appl. Acoust. **45**, 193–210 (1995).
[18] G. R. Moore, "An Approach to the Analysis of Sound in Auditoria," Ph.D. thesis, Cambridge, UK, 1984.
[19] N. Tsingos and J.-D. Gascuel, "Soundtracks for computer animation: Sound rendering in dynamic environments with occlusions," *Graphics Interface'97*, May 1997, pp. 9–16.
[20] M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis* (Academic, New York, 1993).
[21] F. X. Sillion and C. Puech, *Radiosity and Global Illumination* (Morgan Kaufmann, San Francisco, 1994).
[22] K. H. Kuttruff, "Auralization of impulse responses modeled on the basis of ray-tracing results," J. Audio Eng. Soc. **41**(11), 876–880 (1993).
[23] U. R. Kristiansen, A. Krokstad, and T. Follestad, "Extending the image method to higher-order reflections," J. Appl. Acoust. **38**(2-4), 195–206 (1993).
[24] R. Cook, T. Porter, and L. Carpenter, "Distributed ray-tracing," *ACM Computer Graphics, SIGGRAPH'84 Proceedings*, July 1984, Vol. 18(3), pp. 137–146.
[25] J. T. Kajiya, "The rendering equation," *ACM Computer Graphics, SIGGRAPH'86 Proceedings*, Vol. 20(4), pp. 143–150.
[26] H. Lehnert, "Systematic errors of the ray-tracing algorithm," Appl. Acoust. **38**, 207–221 (1993).
[27] N. Dadoun, D. G. Kirkpatrick, and J. P. Walsh, "The geometry of beam tracing," *Proceedings of the Symposium on Computational Geometry*, June 1985, pp. 55–71.
[28] P. Heckbert and P. Hanrahan, "Beam tracing polygonal objects," *ACM Computer Graphics, SIGGRAPH'84 Proceedings*, Vol. 18(3), pp. 119–127.
[29] M. Monks, B. M. Oh, and J. Dorsey, "Acoustic simulation and visualisation using a new unified beam tracing and image source approach," *Proc. Audio Engineering Society Convention*, 1996, pp. 153–174.
[30] U. Stephenson and U. Kristiansen, "Pyramidal beam tracing and time dependent radiosity," *Fifteenth International Congress on Acoustics*, June 1995, pp. 657–660.
[31] J. P. Walsh and N. Dadoun, "What are we waiting for? The development of Godot, II," 103rd Meeting of the Acoustical Society of America, April 1982.
[32] J. H. Chuang and S. A. Cheng, "Computing caustic effects by backward beam tracing," Visual Comput. **11**(3), 156–166 (1995).
[33] A. Fujomoto, "Turbo beam tracing—A physically accurate lighting simulation environment," *Knowledge Based Image Computing Systems*, May 1988, pp. 1–5.
[34] G. Ghazanfarpour and J. M. Hasenfratz, "A beam tracing with precise antialiasing for polyhedral scenes," Comput. Graph. **22**(1), 103–115 (1998).
[35] E. Haines, "Beams O' Light: Confessions of a hacker," *Frontiers in Rendering, Course Notes, SIGGRAPH'91*, 1991.
[36] M. Watt, "Light-water interaction using backward beam tracing," *ACM Computer Graphics, SIGGRAPH'90 Proceedings*, August 1990, pp. 377–385.
[37] C. B. Jones, "A new approach to the 'hidden line' problem," Comput. J. **14**(3), 232–237 (1971).
[38] T. Funkhouser, "A visibility algorithm for hybrid geometry- and image-based modeling and rendering," Comput. Graph. **23**(5), 719–728 (1999).
[39] S. Teller, "Visibility Computations in Densly Occuded Polyhedral Environments," Ph.D. thesis, Computer Science Div., University of California, Berkeley, 1992.
[40] S. Fortune, "Algorithms for prediction of indoor radio propagation," Technical Report Document 11274-960117-03TM, Bell Laboratories, 1996.
[41] S. J. Fortune, "Topological beam tracing," in *Proc. 15th ACM Symposium on Computational Geometry*, 1999, pp. 59–68.
[42] J. Amanatides, "Ray tracing with cones," *ACM Computer Graphics, SIGGRAPH'84 Proceedings*, July 1984, Vol. 18(3), pp. 129–135.
[43] J. P. Vian and D. van Maercke, "Calculation of the room response using a ray tracing method," *Proceedings of the ICA Symposium on Acoustics and Theater Planning for the Performing Arts*, 1986, pp. 74–78.
[44] T. Lewers, "A combined beam tracing and radiant exchange computer model of room acoustics," Appl. Acoust. **38**, 161–178 (1993).
[45] P. Kreuzgruber, P. Unterberger, and R. Gahleitner, "A ray splitting model for indoor radio propagation associated with complex geometries," in *Proceedings of the 1993 43rd IEEE Vehicular Technology Conference*, 1993, pp. 227–230.
[46] A. Rajkumar, B. F. Naylor, F. Feisullin, and L. Rogers, "Predicting RF coverage in large environments using ray-beam tracing and partitioning tree represented geometry," Wireless Networks **2**(2), 143–154 (1996).
[47] S. Teller and P. Hanrahan, "Global visibility algorithms for illumination computations," pp. 239–246 (1993).
[48] S. Teller, C. Fowler, T. Funkhouser, and P. Hanrahan, "Partitioning and ordering large radiosity computations," *ACM Computer Graphics, SIGGRAPH'93 Proceedings*, August 1994, pp. 443–450.
[49] P. Min and T. Funkhouser, "Priority-driven acoustic modeling for virtual environments," *EUROGRAPHICS 2000*, August 2000, pp. 179–188.
[50] F. Durand, G. Drettakis, and C. Puech, "The visibility skeleton: A powerful and efficient multi-purpose global visibility tool," in *Computer Graphics, SIGGRAPH'97 Proceedings*, 1997, pp. 89–100.
[51] B. G. Baumgart, "Winged edge polyhedron representation," Technical Report AIM-179 (CS-TR-74-320), Computer Science Department, Stanford University, Palo Alto, CA, October 1972.
[52] D. P. Dobkin and M. J. Laszlo, "Primitives for the manipulation of three-dimensional subdivisions," Algorithmica **4**(1), 3–32 (1989).
[53] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," *ACM Computer Graphics, SIGGRAPH '80 Proceedings*, July 1980, Vol. 14(3), pp. 124–133.
[54] B. F. Naylor, "Constructing good partitioning trees," *Graphics Interface'93*, May 1993, pp. 181–191.
[55] J. B. Keller, "Geometrical theory of diffraction," J. Opt. Soc. Am. **52**(2), 116–130 (1962).
[56] S. Teller, "Computing the antiumbra cast by an area light source," *ACM Computer Graphics, SIGGRAPH'92 Proceedings*, July 1992, Vol. 26(2), pp. 139–148.
[57] J. Goodman and J. O'Rourke (eds.), *Handbook of Discrete and Computational Geometry* (CRC, Boca Raton, FL, 1997).
[58] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C, 2nd edition* (Cambridge U.P., New York, 1992).
[59] L. Aveneau, Y. Pousset, R. Vauzelle, and M. Méria ux, "Development and evaluations of physical and computer optimizations for the 3d utd model," *AP2000 Millennium Conference on Antennas & Propagation* (Poster), April 2000.
[60] H. Lehnert and J. Blauert, "Principles of binaural room simulation," Appl. Acoust. **36**, 259–291 (1992).
[61] M. E. Delany and E. N. Bazley, "Acoustical characteristics of fibrous absorbent materials," Technical Report NPL AERO REPORT Ac37, National Physical Laboratory, Aerodynamics Division, March 1969.
[62] A. D. Pierce, *Acoustics. An introduction to its physical principles and applications*, 3rd ed. (American Institute of Physics, New York, 1984).
[63] S.-I. Thomasson, "Reflection of waves from a point source by an impedance boundary," J. Acoust. Soc. Am. **59**, 780–785 (1976).
[64] R. G. Kouyoumjian and P. H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface," Proc. IEEE **62**, 1448–1461 (1974).
[65] T. Kawai, "Sound diffraction by a many sided barrier or pillar," J. Sound Vib. **79**(2), 229–242 (1981).
[66] M. A. Biot and I. Tolstoy, "Formulation of wave propagation in infinite media by normal coordinates with an application to diffraction," J. Acoust. Soc. Am. **29**, 381–391 (1957).
[67] H. Medwin, E. Childs, and G. Jebsen, "Impulse studies of double diffraction: A discrete huygens interpretation," J. Acoust. Soc. Am. **72**, 1005–1013 (1982).
[68] U. P. Svensson, R. I. Fred, and J. Vanderkooy, "Analytic secondary source model of edge diffraction impulse responses," J. Acoust. Soc. Am. **106**, 2331–2344 (1999).

[69] R. R. Torres, U. P. Svensson, and M. Kleiner, ''Computation of edge diffraction for more accurate room acoustics auralization,'' J. Acoust. Soc. Am. **109**, 600–610 (2001).
[70] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization* (MIT, Cambridge, MA, 1983).
[71] A. Stettner and D. P. Greenberg, ''Computer graphics visualization for acoustic simulation,'' *ACM Computer Graphics, SIGGRAPH'89 Proceedings*, July 1989, Vol. 23(3), pp. 195–206.
[72] J. Arvo and D. Kirk, ''A survey of ray tracing acceleration techniques,'' *An Introduction to Ray Tracing*, 1989.
[73] C. Sequin and E. Smyrl, ''Parameterized ray tracing,'' *ACM Computer Graphics, SIGGRAPH'89 Proceedings*, July 1989, Vol. 23(3), pp. 307–314.
[74] M. Monks, B. M. Oh, and J. Dorsey, ''Audioptimization: Goal based acoustic design,'' *IEEE Computer Graphics & Applications*, May 2000, pp. 76–91.
[75] M. Vorlander, ''International round robin on room acoustical computer simulations,'' in *Proceedings of the 15th International Congress of Acoustics*, June 1995.
[76] K. Nakagawa, T. Miyajima, and Y. Tahara, ''An improved geometrical sound field analysis in rooms using scattered sound and an audible room acoustic simulator,'' J. Appl. Acoust. **38**(2-4), 115–130 (1993).
[77] G. M. Naylor and J. H. Rindel, ''Predicting room acoustical behavior with the odeon computer model,'' in *Proceedings of the 124th ASA Meeting*, November 1992, p. 3aAA3.