Real-Time Acoustic Modeling for Distributed Virtual Environments

Thomas Funkhouser
Princeton University

Patrick Min
Princeton University

Ingrid Carlbom
Bell Laboratories

Abstract

Realistic acoustic modeling is essential for spatializing sound in distributed virtual environments where multiple networked users move around and interact visually and aurally in a shared virtual world. Unfortunately, current methods for computing accurate acoustical models are not fast enough for real-time auralization of sounds for simultaneously moving sources and receivers. In this paper, we present three new beam tracing algorithms that greatly accelerate computation of reverberation paths in a distributed virtual environment by taking advantage of the fact that sounds can only be generated or heard at the positions of "avatars" representing the users. The *priority-driven beam tracing* algorithm performs a best-first search of a cell adjacency graph, and thus enables new termination criteria with which all early reflection paths can be found very efficiently. The *bidirectional beam tracing* algorithm combines sets of beams traced from pairs of avatar locations to find reverberation paths between them while requiring significantly less computation than previous unidirectional algorithms. The *amortized beam tracing* algorithm computes beams emanating from box-shaped regions of space containing predicted avatar locations and re-uses those beams multiple times to compute reflections paths as each avatar moves inside the box. Cumulatively, these algorithms enable speedups of approximately two orders of magnitude over previous methods. They are incorporated into a time-critical multiprocessing system that allocates its computational resources dynamically in order to compute the highest priority reverberation paths between moving avatar locations *in real-time* with graceful degradation and adaptive refinement.

Key Words: Virtual environment systems, virtual reality, acoustic modeling, auralization, beam tracing.

1 Introduction

Distributed virtual environment (DVE) systems incorporate computer graphics, sound, and networking to simulate the experience of real-time interaction between multiple users represented by avatars in a shared three-dimensional virtual world. They allow a user to "explore" information and "interact" with other users in the context of a virtual environment by rendering images and sounds of the environment in real-time while the user "moves" through the 3D environment interactively. Example applications for DVE systems include collaborative design [4], distributed training [35], teleconferencing [29], and multi-player games [27].

A difficult challenge in implementing a DVE system is to render realistic sounds spatialized according to the virtual environment in real-time on every participating user's computer. Sound waves originating at a source location travel through the environment along a multitude of reverberation paths, representing different sequences of reflections, transmissions, and diffractions. The different arrival times and amplitudes of sound waves traveling along these paths provide important auditory cues for localization of objects, separation of simultaneous speakers (i.e., the "cocktail party effect"), and sense of presence in a virtual environment [11].

The goal of our work is to build a DVE system in which multiple users can communicate with each other in a 3D virtual world with realistic spatialized sound. User generated sounds and avatar movements are transmitted via network messages to an audio server which spatializes sounds according to impulse responses encoding the reverberations computed for every pair of avatars (see Figure 1). The new research challenge is to develop geometric acoustic modeling algorithms to compute reverberation paths between every pair of avatar locations *in real-time* as they move through a 3D environment.

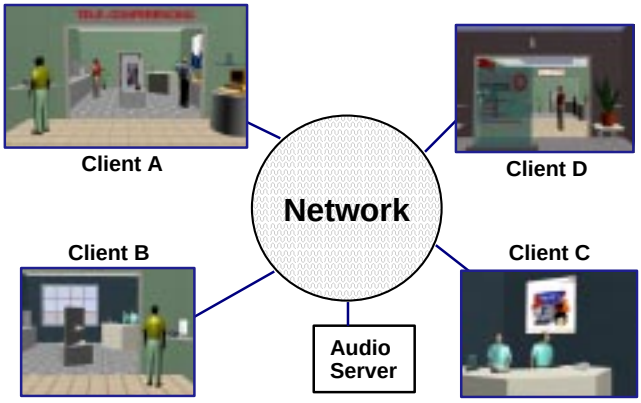


Figure 1: Distributed virtual environment system.

Acoustic modeling algorithms for DVE systems must be efficient, adaptive, predictive, and asynchronous so that appropriate impulse responses are ready for use without delay as avatars move. Since the numbers and locations of avatars inhabiting the virtual environment can be highly variable, it is not practical to preallocate a fixed amount of computational resources to the update of reverberation paths between any pair of avatars. Some avatars may be moving very closely to one another within the same room, requiring impulse response updates at more than 10Hz to avoid noticeable localization artifacts, while others may be separated by great distances and require less frequent and/or less accurate acoustic simulations (people can detect angular differences in sound source locations separated by 5 degrees [3]). Similarly, the quality of the computed results should degrade gracefully as the computational demands increase (e.g., when more users inhabit the environment), and it should refine adaptively as more resources become available (e.g., when all the users remain stationary). Time critical algorithms must be used to guarantee update rates and allocate computational resources effectively and adaptively so that the overall quality of computed impulse responses is the best possible.

In this paper, we describe geometric beam tracing algorithms to compute reverberation paths between moving avatar locations at very high rates. The primary research contributions are embodied in the three methods we introduce for accelerating computation of reverberation paths: 1) priority-driven beam tracing, 2) bidirectional beam tracing, and 3) amortized beam tracing. We have integrated these methods into a time-critical computing framework to build a complete DVE system incorporating realistic imagery and spatialized sound for real-time communication between multiple networked users.

2 Previous Work

There has been decades of work in *off-line* acoustic modeling for applications such as concert hall design. Previous methods include path tracing [22], image source methods [1, 5], boundary element methods [26, 33], and beam tracing [8, 18]. Surveys of work in this area appear in [3, 21, 23]. In general, current off-line systems compute reverberation paths for a small set of pre-specified source and receiver locations, and they allow interactive evaluation only for precomputed results. Unfortunately, it is usually not possible to store precomputed impulse responses or reverberation paths over all possible avatar locations for use by a distributed virtual environment system, as the storage requirements of this approach would be prohibitive for all cases except very simple environments or very coarse samplings.

There have also been many advances over the last decade in distributed virtual environments systems supporting visual interactions between networked users in a shared 3D virtual environment (an early example is Reality Built for Two [4]). The most common examples include multi-player games (e.g., Quake [27]), military battle simulations (e.g., NPSNET [35]), and multi-user chat environments (e.g., Sony's Community Place [29]). These programs display images in real-time with complex global illumination and textures to produce visually compelling immersive experiences.

On the other hand, there has been relatively little progress in real-time acoustic modeling. Current on-line systems generally consider only simple geometric arrangements and low-order specular reflections. For instance, the Acoustetron [13] computes only first- and second-order specular reflections for box-shaped virtual environments, while video games provide spatialized sound with ad hoc localization methods (e.g., pan effects), rather than with realistic acoustic modeling methods. Almost all DVE systems attenuate sound with distance, and many support user-specified "regions of influence" (e.g., ellipsoids) for each sound source [16]. However, to quote the 1995 National Research Council Report on Virtual Reality Scientific and Technological Challenges [12], "current technology is still unable to provide interactive systems with real-time rendering of acoustic environments with complex, realistic room reflections."

The new algorithms presented in this paper are based on the beam tracing method described in [14]. To review, beams are traced from the position of each *stationary* sound source along paths of transmission and specular reflection via a depth-first traversal of an adjacency graph of polyhedral cells (see Figure 2). The algorithm starts in the cell containing the source with a beam representing the entire cell (labeled 'D'). Then, it recursively traces convex pyramidal beams through intersected cell boundaries into adjacent cells, incrementally trimming the beams by convex polygons at traversed cell boundaries (u, o, p, t, and s), and mirroring the beams at reflecting cell boundaries (o and p). The recursion of each depth-first traversal is captured and stored in a beam tree, representing all the traced reverberation paths emanating from the source. Later, during an interactive session, the precomputed beam trees are used to generate reverberation paths to a moving receiver position. For every beam containing the receiver, a reverberation path is constructed by iterative intersection with the reflecting cell boundaries stored with the ancestors of the corresponding beam tree node (as shown in Figure 2).

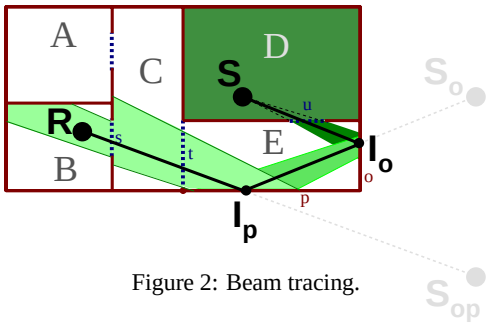


Figure 2: Beam tracing.

The key idea in this previous method is that it is possible to use an off-line precomputation to construct a data structure (a beam tree) encoding potential reverberation paths from each static source location and use that precomputed data structure to compute reverberation paths to a moving receiver quickly for auralization. Since the off-line beam tracing algorithm takes up to 50 seconds for each new source location [14], this method is not directly applicable for computing reverberation paths in real-time between *moving* avatars in a distributed virtual environment. Other data structures proposed for interactive updates of global illumination solutions are best suited for handling changes to materials and geometry in primarily diffuse environments [6, 10], and they would generally not perform well for viewpoint changes in highly specular environments.

The focus of our new work is developing beam tracing algorithms that can be used to compute reverberation paths *in real-time* with update rates suitable for auralization in distributed virtual environment applications (up to 100Hz). Achieving this goal requires acceleration of current beam tracing methods by two or three orders of magnitude.

3 Overview of Approach

Our approach is to take advantage of the fact that DVE systems must only compute reverberation paths between avatar locations,[1] and not between all points in 3D space. Since avatars are located at a discrete set of positions, and they tend to move along continuous paths, beam tracing algorithms for DVEs can utilize directed searches and temporal coherence to find reverberation paths very efficiently. In this paper, we propose the following three beam tracing algorithms that exploit these properties:

- **Priority-driven beam tracing** uses psychoacoustically motivated priorities and termination criteria to trace only beams that can represent the most significant reverberation paths from one avatar location to another.
- **Bidirectional beam tracing** combines two sets of beams traced from different avatar locations to find reverberation paths efficiently.
- **Amortized beam tracing** utilizes beams traced from a region of space to compute reverberation paths quickly for a sequence of nearby avatar locations.

The following three sections contain detailed descriptions of these new algorithms, while Sections 7 and 8 describe how they are integrated into our adaptive, real-time DVE system.

[1] We extend the notion of an avatar to include anything that can be a source or receiver of sound.

4 Priority-Driven Beam Tracing

The first algorithm is motivated by priority-driven search methods. For a given avatar location, all reverberation paths are not equally important. Some paths are psychoacoustically very significant (e.g., direct paths to other avatars), while others follow complex sequences of reflections towards empty regions of space.

Our priority-driven beam tracing algorithm exploits knowledge of avatar locations to compute only the most significant reverberation paths efficiently. Specifically, the algorithm considers beams in *best-first* order, rather than depth-first order as in previous beam tracing algorithms (e.g., [14]). As beam trees are constructed, the leaf nodes are stored in a priority queue, and the highest priority node is iteratively popped for expansion at each step. The benefit of this approach is that the most significant beams are considered first, enabling methods based on adaptive refinement and dynamic termination criteria.

The first issue in implementing the priority-driven algorithm is to assign relative priorities to different beam tree nodes. Following accepted practice of the acoustics literature, we partition the reverberation paths represented by the beams into two categories: 1) early reflections, and 2) late reverberations. Early reflections are defined as the ones that arrive at the receiver within some short amount of time, T_e, after the most direct sound, while late reverberations comprise the rest ($20ms \leq T_e \leq 80ms$ [3, 17]). Since localization in the human brain is most sensitive to the early reflections, and the late reverberations are characterized by many paths arriving from all directions after being multiply scattered by many surfaces, geometric acoustic modeling systems generally compute only early specular reflections, while late reverberations and diffractions are modeled with statistical approximations (see Figure 3). In our current system, we use this approach, assigning higher priorities to beam tree nodes representing shorter reflection paths.

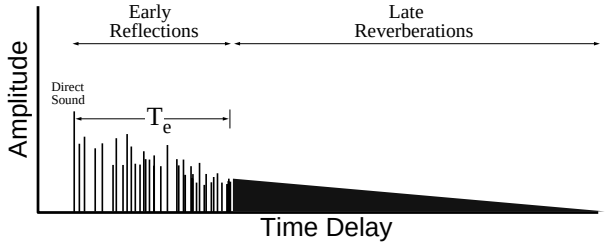


Figure 3: Impulse response.

The second issue is how to guide our priority-driven algorithm to find early specular reflection paths efficiently. Previous algorithms have traced beams in depth-first order up to some termination criteria based on the maximum number of reflections, maximum path length, and/or maximum attenuation. One problem with this approach is that it is impossible to select a priori a termination criterion that guarantees finding all early specular reflection paths. In general, the length of the longest early reflection path cannot be predetermined, since it depends on the length of the shortest one, which is not known until it is found.

Priority-ordered search helps us overcome this problem. We assign the priority , $f(B)$, of each beam tree node, B, to the length of the shortest path from the beam source to the last traversed cell boundary, $g(B)$, plus the length of the shortest path from that cell boundary to the closest avatar location, $h(B)$ (see Figure 4). Since $f(B)$ underestimates the length of any path through B to an avatar, we can be assured that all early specular reflection paths are found if we terminate the traversal when the value of $f(B)$ for all nodes remaining in the priority queue corresponds to an arrival time at least T_e later than the most direct path found to every avatar location.

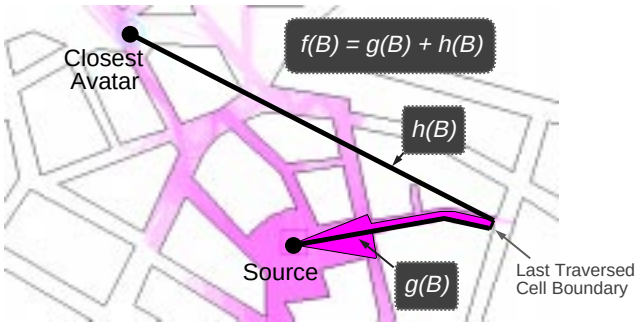


Figure 4: The priority heuristic, $f(B)$, underestimates the distance traveled so far, $g(B)$, plus the distance to the closest avatar, $h(B)$.

Our implementation of this approach is closely related to the classical A^* algorithm from the artificial intelligence literature [15]. The difference is that the search for each avatar locations has multiple goals (i.e., the early reflection paths to all other avatar locations). Thus, we must remember the shortest path to each avatar when we find it, and we must remember for which avatars we have already considered all potential early reflection paths.

The primary advantage of the priority-driven beam tracing method is that it avoids geometric computations for many beams representing insignificant reverberation paths, and therefore it is able to compute the significant ones more rapidly. It can also be used to allocate resources dynamically in a real-time system (see Section 7). The disadvantage is that there is extra overhead in computing priorities and maintaining priority queues of beam tree nodes.

5 Bidirectional Beam Tracing

The second algorithm uses a bidirectional approach to combine beam trees traced independently from two different avatar locations to find reverberation paths between them.

The primary motivation for a bidirectional approach is that the computational complexity of beam tracing algorithms grows exponentially with increasing reflections. Consequently, tracing one set of beams up to k reflections usually takes far longer than tracing two sets of beams up to $k/2$ reflections.

A second motivation is that DVE systems must find reverberation paths between all pairs of avatars. In this situation, unidirectional approaches are inherently redundant, since beams must be traced fully from all except one avatar locations to insure that reverberation paths are found between all pairs. But then almost every reverberation path is traced twice, once in each direction. With a bidirectional approach, we can avoid this redundant work by combining beams traced from one avatar location with beams traced from another to find the same reverberation paths more efficiently.

Bidirectional path tracing approaches have been investigated for decades in ray tracing (e.g., [7, 24, 34]), radiosity (e.g., [20, 28]), and other fields (e.g., [9, 25]). Generally, for every ray found impinging upon a surface, a data structure associated with the surface is both updated and queried to compute the energy traveling forwards and backwards along the ray. The main challenge of the bidirectional approach is to find a data structure that efficiently and accurately represents the directional energy radiating from every surface of the environment. Previous data structures for bidirectional path tracing have mostly been based on storing discrete radiance samples (e.g., "illumination maps" [2] and "radiosity textures" [19]), and thus they suffer from aliasing artifacts. An important observation is that beam tree data structures [18] are an object-space

representation of the directional energy radiating from each surface, and they are well-suited for implementing a bidirectional approach.

Our bidirectional beam tracing algorithm computes reverberation paths by combining beam trees traced independently from different avatar locations. The key contribution of our algorithm is the method we use for determining which beams, B_1 and B_2, traced independently from avatar locations, P_1 and P_2, combine to represent viable reverberation paths. We base our method on the following observations, which apply for common ray propagation models comprising specular reflections, diffuse reflections, transmissions, and diffractions over locally reacting surfaces.[2]

- **Condition A:** There is a reverberation path if B_1 contains P_2 (see Figure 5a).
- **Condition B:** There are (usually an infinite number of) reverberation paths containing a diffuse reflection at surface S if both B_1 and B_2 intersect the same region of S (see Figure 5b).
- **Condition C:** There is a reverberation path containing a straight-line transmission through surface S if: 1) both B_1 and B_2 intersect the same region of S, 2) B_1 intersects the virtual source of B_2, and 3) B_2 intersects the virtual source of B_1 (see Figure 5c).
- **Condition D:** There is a reverberation path containing a specular reflection at surface S if: 1) both B_1 and B_2 intersect the same region of S, 2) B_1 intersects the mirrored virtual source of B_2, and 3) B_2 intersects the mirrored virtual source of B_1 (see Figure 5d).
- **Condition E:** There is a reverberation path containing a diffraction at an edge E if: 1) B_1 and B_2 both intersect the same region of E (see Figure 5e).

To accelerate determination of these conditions, we construct lists of beam tree nodes intersecting each cell and face of the spatial subdivision as the beams are traced. We traverse these lists to determine efficiently which pairs of beam tree nodes potentially combine to represent viable reverberation paths, avoiding consideration of all $n(n-1)/2$ pairwise combinations of traced beams. First, for each pair of beam tree nodes considered, we check to see if both nodes are either the root or a leaf of their respective beam trees. If not, the pair can be ignored, as it surely represents a reverberation path that will be found by another pair of nodes. Otherwise, we check the beams intersecting each cell containing an avatar to see if they satisfy Condition A. We check pairs of beams intersecting the same transmissive face to see if they satisfy Condition C. We check pairs of beams intersecting the same reflecting face to see if they satisfy Conditions D. Finally, we use the first node meeting one of these criteria to compute an underestimating distance heuristic to another avatar location, which can be used to aid early termination when searching for early reflection paths in an integrated bidirectional and priority-driven beam tracing algorithm.

As compared to unidirectional beam tracing methods, the main advantage of our bidirectional algorithm is that paths with up to R reflections can be found by combining two beam trees representing up to R_1 and R_2 reflections, respectively, where $R_1 + R_2 - 1 = R$. Since $c^{R_1} + c^{R_2} << c^R$ for most c, fewer beams must be traced. The main disadvantage is that extra bookkeeping and processing is required to combine beams from different beam trees.

[2] So far, we have implemented the methods only for specular reflections.

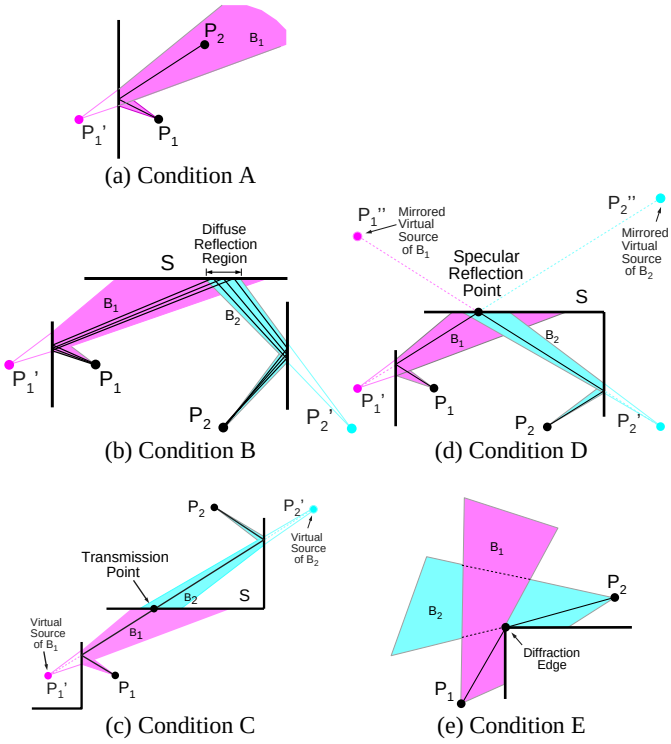


Figure 5: Combining bidirectional beams.

6 Amortized Beam Tracing

The third algorithm accelerates computation of reflection paths between moving avatars by amortizing the cost of a single beam tracing computation over several receiver locations.

The paths taken by people as they move through a virtual environment generally exhibit a large amount of spatial and temporal coherence. People usually do not jump from point to point arbitrarily. Rather, they tend to move along continuous paths. User interfaces may even impose bounds on their speeds, rotations, and directions of movement. Moreover, a person's two ears are nearby one another at fixed relative positions. By utilizing temporal coherence and spatial constraints we can model reverberation paths more efficiently for a sequence of receiver locations than for a set of unrelated locations. Specifically, we extend the notion of a beam tree to include *conservative beams* traced from a region of space, rather than just a point. Since these beams over-estimate the set of reverberation paths emanating from any point within the region, we can usually re-use them for a sequence of receiver locations inside the region to identify potential reverberation paths more rapidly than tracing beams from scratch for every avatar movement.

Our approach is motivated by algorithms originally proposed for conservative visibility determination (visibility is tracing 0th-order paths) [30, 32]. These algorithms do not solve the visibility problem exactly. Instead, they simply reduce the size of the polygon set to be processed by a later hidden-surface algorithm (e.g., z-buffer). In the same way, a conservative beam tree does not exactly represent the reflection paths for any point. Instead, it encodes a superset of the possible sequences of reflections and transmissions from all points within its source region. Thus, to find exact reflection paths from such a point to another point, we must only check a relatively small set of potential polygon sequences encoded in the beam tree to see which of them admits a "valid" reflection path. These relatively simple checks can be executed very quickly for each pair of avatar locations, while the much longer time required for beam tracing is

amortized over multiple avatar movements.

Our amortized beam tracing algorithm is integrated with the priority-driven and bidirectional methods as shown in Figure 6. From top to bottom, we first use motion prediction to update a set of *source regions* that we expect to contain future avatar locations. Then, in the second step, we trace conservatively over-estimating polyhedral beams in priority order from each source region and store them in a *conservative beam tree*. Next, for every pair of source regions, we use our bidirectional methods to combine the conservative beam trees to form a set of *polygon sequences*, each of which describes how a ray traveling from one region to the other can potentially reflect off and/or transmit through an ordered list of polygons. Finally, for each time an avatar moves within one of the source regions, we generate exact reflection paths to every other avatar simply by processing the appropriate set of polygon sequences. This last step is the only one repeated for every avatar movement, and it is usually very fast.

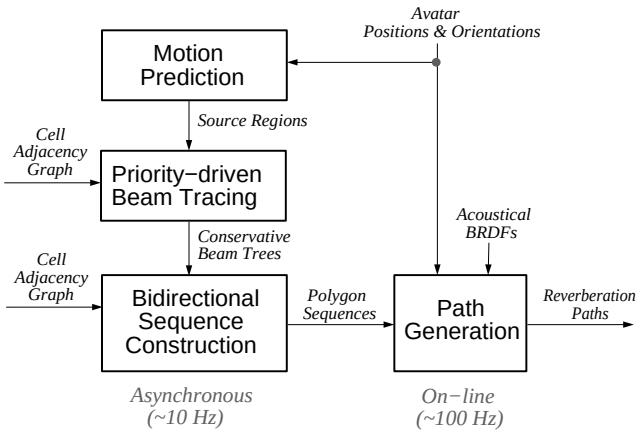


Figure 6: Amortized beam tracing phases.

The key challenges in implementing this algorithm are: 1) selecting effective source regions, 2) efficiently tracing a set of tightly over-estimating beams emanating from a region of space, 3) forming polygon sequences by combining beams traced from two regions of space, and 4) checking polygon sequences quickly to see if they admit valid reflection paths for a specific pair of avatar locations.

First, to select effective source regions, we must balance competing goals. Each one should contain as many future avatar locations as possible so that its conservative beam tree can be re-used many times. However, they cannot be too large (e.g., a whole room). Otherwise, the beams traced from the region are "too over-estimating," which not only slows down the beam tracing computation, but also puts undue processing burden on the subsequent steps. Our current method updates a source region associated with each avatar by predicting its motion with a second-degree polynomial curve and detecting collisions with obstacles in the environment. The source region is always an axis-aligned box with user-specified dimensions ("Box Size"). We are currently extending this method to adapt the size of each source region according to the locations of other avatars and the geometric complexity of the local environment.

Second, to trace beams from a box-shaped source region efficiently, we over-estimate the set of paths emanating from the source by a polytope comprising the intersection of at most six halfspaces, with two halfspaces bounded by planes parallel to each of the X, Y, and Z axes. Each of the three pairs of halfspaces form a wedge bounding the beam in one dimension, and the intersection of the three wedges form a polytope over-estimating the potential reverberation paths from any point within the source region. As cell boundaries are traversed in the beam tracing algorithm, the halfspaces are updated incrementally by a gift-wrapping step, in which the plane bounding each halfspace is rotated around an extremal edge of the source box until it hits an extremal vertex of the cell boundary, as shown in Figure 7. Although this beam tracing method is slightly more over-estimating than previously described approximations (e.g., [32]), it is very fast, requiring constant time per cell boundary, and it provides a fairly tight over-estimate for the paths traced in typical virtual environments.

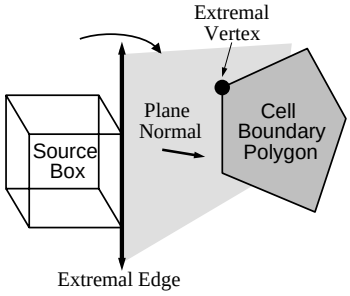


Figure 7: Conservative beam tracing gift wrapping step.

Third, to form polygon sequences, we extend the bidirectional method described in Section 5 to combine beams whose sources are boxes rather than points and to construct polygon sequences rather than specific reverberation paths. In particular, for each pair of beam tree nodes meeting one of the conditions, A-E, listed in Section 5 (with beam sources extended to include axis-aligned boxes), we construct a polygon sequence by concatenating the ordered lists of reflections and transmissions at cell boundaries encoded in the ancestor nodes of the two beam trees, as shown in Figure 8. Every polygon sequence constructed in this way provides a recipe for how a ray potentially can travel from a point in one source region to a point in the other.

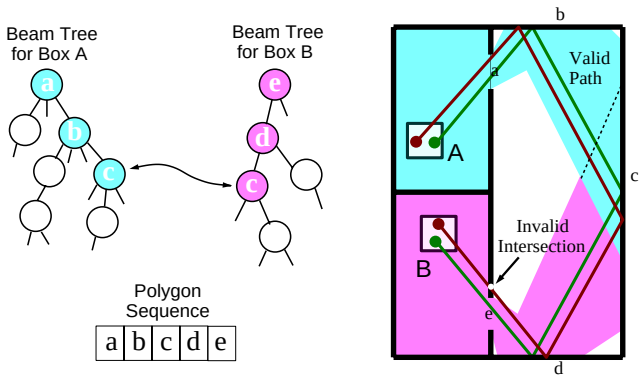


Figure 8: Constructing a polygon sequence by linking beam tree nodes, and determining its validity for specific pairs of avatar locations by checking polygon intersections.

Finally, to find paths between a specific pair of avatar locations, P_1 and P_2, we simply check each polygon sequence, S, associated with the appropriate pair of source regions to see whether a ray traveling from P_1 traveling along the prescribed sequence of reflections and transmissions can possibly reach P_2. We first traverse the polygon sequence in backwards order to construct a stack of mirror images of P_2, where P_2^i corresponds to the image resulting from mirroring P_2 over the last i of the n reflecting polygons in the sequence. Then, we construct the reverberation path by traversing the polygon sequence in forwards order, while iteratively checking

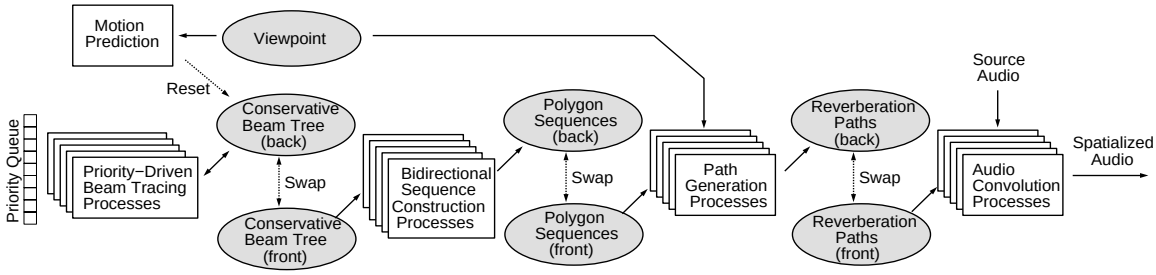


Figure 9: Multiprocessing system organization.

to see whether the ray starting from P_1^i to P_2^{n-i} intersects every polygon, S_i, in the sequence (see Figure 8). If so, we have found a valid reflection path from P_1 to P_2 along S (shown as a green line in Figure 8). The length of the path is given by the distance from P_1 to P_2^{n-1}, and the attenuation of sound traveling along the path is given by the product of the attenuations at each polygon in the sequence. Otherwise, if a ray fails to intersect one of the polygons (e.g., the red line in Figure 8), the sequence is determined to be in the over-estimating portion of the conservative beams, and it is ignored.

One advantage of conservative beam tracing is that the cost of computing each beam can be amortized over all points within its source region. However, tracing beams from a larger region of space is generally more expensive. So, the sizes of source regions must be chosen carefully to maximize the effective speedups. Another advantage is that beam trees and polygon sequences can be constructed asynchronously (or even off-line), and then reflection paths can be found at very high rates. This feature is exploited in the multiprocessing system described in the next section.

7 Time-Critical Multiprocessing

Motivated by the principles of time-critical computing, we have developed an adaptive, real-time multiprocessing system that uses multiple asynchronously executing processes and allocates computational resources dynamically in order to perform the highest priority beam tracing computations in a timely manner. We expand the notion of priority-driven beam tracing to include multiple processors working together on the beam trees for all avatars at once. Rather than updating the beam tree for each avatar independently, the system computes beam tree nodes in global priority order, and thus the system can trade-off computational resources between different avatars dynamically. For instance, if many avatars enter the virtual environment, available resources for computing beam tree nodes for each avatar decreases, and the quality of computed reverberation paths degrades gracefully. Conversely, when avatars leave the environment or become stationary, the quality of spatialized sounds is adaptively refined. In the limit, if there is just one avatar in the environment, all available processors concurrently expand its beam tree nodes in priority order.

Our multiprocessor implementation executes as shown in Figure 9. Each data structure (oval) is stored as a pair of "front" and "back" buffers. Just as in computer graphics, the back buffer is used for updates, while other processes access the front buffer. Updates to all data structures are synchronized by mutual exclusion locks that allow multiple readers, but only a single writer. As a result, a high degree of concurrent processing is possible, as mutual exclusion is required for front buffers only when the buffers are swapped.

As every avatar moves through the environment, a motion prediction process determines when to swap an avatar's beam trees (e.g., when it has moved significantly). When the beam trees are swapped, the new "back" beam tree is emptied and then prepared for expansion: a new source region is assigned and a root node is created and put on the priority queue. Asynchronously and continuously, multiple beam tracing processes work on expanding "back" beam tree nodes in global priority order. Each process iteratively pops a node from the priority queue, computes the children of the node, and pushes them onto the priority queue. Meanwhile, multiple bidirectional polygon sequence construction processes monitor the arrival of new beam trees in the front buffer of any avatar. When one arrives, the "front" and "back" buffers of the appropriate polygon sequences are swapped, and new "back" buffers of polygon sequences are updated from the new beam tree. Similarly, each time an avatar moves, or a new polygon sequence becomes available, path generation processes access the polygon sequences to form reverberation paths. Finally, source audio signals are spatialized via software convolution with binaural impulse responses computed for each avatar. The responses are derived by adding a pulse for each computed reverberation path, where the delay is determined by L/C, where L is the length of the path, and C is the speed of sound, and the amplitude is set to $1/2(1+\cos\theta)A/L$, where θ is the angle of arrival of the pulse with respect to the normal vector pointing out of the ear, and A is the product of all the frequency-independent reflectivity and transmission coefficients for each of the reflecting and transmitting surfaces along the corresponding reverberation path.

The nicest feature of this implementation is that it provides a framework in which each component of the system can "do the best it can" without slowing down the other components. In particular, the path generation processes continue to compute reverberation paths even when an avatar moves out of its beam tree's source region. This feature provides "extrapolation" of reverberation paths from nearby points and is critical to providing a seamless auditory experience when the beam tracing processes become overloaded.

8 Experimental Results

We have implemented the algorithms described in the preceding sections in C++ and integrated them into a DVE system supporting communication between multiple users in a virtual world with spatialized sound. Our current implementation supports specular reflections and transmissions in 3D polygonal environments, and it runs on PCs and SGIs connected by a 100Mb/s TCP network.

The system uses a client-server design. Each client provides an immersive audio/visual interface to the shared virtual environment from the perspective of one avatar. As the avatar "moves" through the environment, possibly under interactive user control, images and sounds representing the virtual environment from the avatar's simulated viewpoint are updated on the client computer in real-time. Communication between remote users on different clients is possible via network connections to the server(s). Any client can send messages to the server(s) describing updates to the environment (e.g., the position and orientation of avatars) and the sounds occurring in the environment (e.g., voices associated with

avatars). When a server receives these messages, it processes them to determine which updates are relevant to which clients, it spatializes the sounds for all avatars with the beam tracing algorithms described in the preceding sections, and it sends appropriate messages with updates and spatialized audio streams back to the clients so that they may update their audio/visual displays.

To evaluate the effectiveness of our new beam tracing methods in the context of this system, we executed a series of experiments with a single server spatializing sounds on an SGI Onyx2 with four 195MHz R10000 processors. In each experiment, we used different beam tracing algorithms to compute specular reflection paths from a source point (labeled 'A') to each of the three receiver points labeled 'B,' 'C,' and 'D' in the 3D model shown in Figure 10.

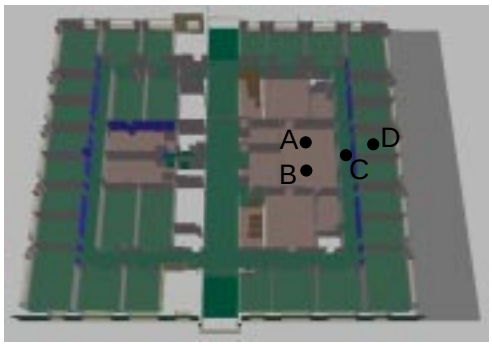


Figure 10: Test model geometry and avatar locations.

8.1 Priority-Driven Beam Tracing Results

We tested the relative benefits and costs of priority driven beam tracing by running a series of tests using the following three different beam tracing algorithms based on different search methods for traversing the cell adjacency graph and different termination criteria:

- **DF-R:** Depth-first search up to a user-specified maximum number of reflections.
- **DF-L:** Depth-first search up to a user-specified maximum path length.
- **P:** Priority-driven search (our algorithm).

In each set of tests, we computed all early specular reflection paths (T_e = 20ms) from a source point (labeled 'A') to one of three receiver points (labeled 'B,' 'C,' and 'D') in the 3D model shown in Figure 10. The depth-first search algorithms, DF-R and DF-L, were aided by oracles in these tests, as the termination criteria were chosen manually to match the exact maximum number of reflections, R, and the maximum path length, L, respectively, of known early reflection paths, which were predetermined in earlier tests. In contrast, the priority-driven algorithm, P, was given no hints, and it used only the dynamic termination criteria described in Section 4.

The bar chart in Figure 11 shows the wall-clock times (in seconds) required to find all early specular reflection paths for each combination of the three receiver points and the three beam tracing algorithms. Although all three algorithms find exactly the same set of early reflection paths from the source to each receiver, the computation times for the priority-driven approach (the blue bars) were between 2.6 and 4.3 times less than the next best. The reason is that the priority-driven algorithm considers beams representing earliest paths first and terminates according to a metric utilizing knowledge of the receiver location, and thus it avoids computing most of the useless beams that travel long distances from the source and/or stray far from the receiver location.

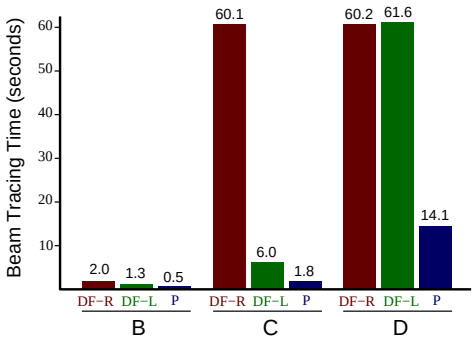


Figure 11: Beam tracing times with different traversal algorithms and termination criteria.

The relative value of the priority-driven approach depends on the geometric properties of the environment. For instance, all early reflection paths to the receiver point 'B,' which was placed in the same room as the source, required less than or equal to 3 specular reflections, and the longest path was only 623 inches. These relatively tight termination criteria were able to bound the complexities of the depth first search algorithms, so the speedup of the priority-driven algorithm is only around 2.6x over the next best. In contrast, for receiver point 'D,' some early reflection paths required up to 7 specular reflections, and the longest early reflection path was 1046 inches. In this case, the priority-driven algorithm is far more efficient (speedup is 4.3x) as it directs the beam tracing search towards the receiver point almost immediately, rather than computing beams extending radially in all directions.

The benefits of directed search are shown visually in Figure 12, which contains two images of 10,000 beams (pink) traced in a 2D map of Boston from a source point (square) to two different receiver locations (circles). In this case, the distance heuristic directs beams towards each receiver location enabling it to find complex reflection paths (blue).

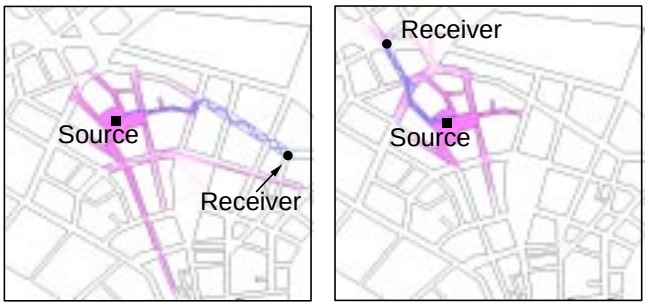


Figure 12: Visualization of beams traced for different receiver points with priority-driven beam tracing.

8.2 Bidirectional Beam Tracing Results

To test the relative benefits and costs of the bidirectional beam tracing algorithm described in Section 5, we ran a series of tests with comparable unidirectional and bidirectional beam tracing implementations on an SGI workstation with a 195MHz R10000 processor.

In each set of tests, we computed all specular reflection paths from a source point (labeled 'A') to one of three receiver points

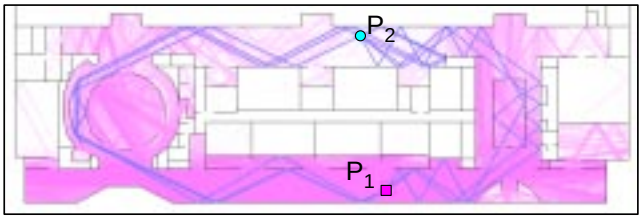
(labeled 'B,' 'C,' and 'D') up to a specified maximum number of reflections ('R') in the 3D model shown in Figure 10. The unidirectional algorithm constructed a single beam tree containing all paths with up to R specular reflections from the source point, and then it reported a specular reflection path for each beam containing the specified receiver point. In contrast, the bidirectional algorithm constructed two beam trees for each source-receiver pair, the first containing beams up to $R/2+1$ specular reflections from the source, and the second containing beams up to $R/2$ specular reflections from the receiver. The two beam trees were combined to find all Rth-order specular reflections. In this experiment, the unidirectional and bidirectional algorithms find exactly the same set of paths with up to R specular reflections. The goal of the experiment is to determine which algorithm takes less total computation time.

Table 1 contains statistics collected during these tests. From left to right, the first column (labeled 'P') lists which receiver point was used. The second column (labeled 'R') indicates the maximum number of specular reflections computed. Then, for both the unidirectional and bidirectional algorithms, there are three columns which show the times (in seconds) required to compute the beam trees ("Beam Time"), find the reflection paths ("Path Time"), and the sum of these two ("Total Time"). Finally, the last column (labeled "Speedup") lists the total time for unidirectional beam tracing algorithm as a ratio over the total bidirectional beam tracing time.

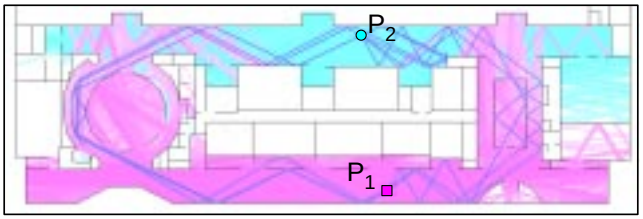
		Unidirectional			Bidirectional			
P	R	Beam Time	Path Time	Total Time	Beam Time	Path Time	Total Time	Speed Up
B	3	2.02	0.01	2.03	1.04	0.03	1.07	1.9
	4	5.79	0.03	5.82	2.55	0.10	2.65	2.3
	5	15.01	0.07	15.08	4.23	0.50	4.73	3.5
	6	31.53	0.14	31.66	8.02	1.31	9.33	3.9
	7	60.26	0.24	60.50	11.95	4.43	16.39	5.0
	8	100.82	0.41	101.22	21.12	9.52	30.64	4.8
C	3	2.03	0.01	2.03	0.96	0.01	0.98	2.1
	4	5.81	0.01	5.82	2.49	0.04	2.54	2.3
	5	14.83	0.02	14.86	3.92	0.20	4.12	3.8
	6	31.38	0.05	31.42	7.82	0.54	8.37	4.0
	7	60.82	0.08	60.90	11.23	1.97	13.20	5.4
	8	100.89	0.14	101.03	20.56	4.13	24.69	4.9
D	3	2.03	0.00	2.03	0.62	0.01	0.62	3.3
	4	5.81	0.01	5.81	2.17	0.03	2.20	2.7
	5	14.94	0.01	14.95	2.47	0.12	2.59	6.0
	6	31.88	0.02	31.90	6.24	0.29	6.53	5.1
	7	60.31	0.04	60.35	7.10	0.92	8.02	8.5
	8	100.68	0.06	100.75	16.27	1.83	18.10	6.2

Table 1: Bidirectional beam tracing statistics.

The results of this experiment show the trade-offs of the bidirectional approach very clearly. First, comparing the "Beam Times" in Table 1, we see that the bidirectional algorithm spends significantly less time tracing beams than the unidirectional algorithm. This is because it constructs beam trees with less depth, thereby avoiding the worst part of the exponential growth. A visualization of this effect appears in Figure 13, which shows a comparison of 2D beams traced (a) unidirectionally and (b) bidirectionally to find the same reflection paths (shown in blue). Note how many more beams are traced in the unidirectional case. Second, comparing the "Path Times," we see that the bidirectional algorithm spends significantly more time generating reverberation paths. This is because it must perform intersection tests for each pair of beams reflecting off the same surface. Overall, the bidirectional beam tracing algorithm is significantly faster than the unidirectional algorithm, with speedups ranging from 4.8x to 6.2x for 8 specular reflections in these experiments.



(a) Unidirectional
(59,069 beams)



(b) Bidirectional
(15,078 pink beams + 24,665 blue beams)

Figure 13: Visualization of beams traced to find specular reflection paths between points P1 and P2 with (a) unidirectional and (b) bidirectional methods. The beams emanating from location P1 are shown in pink, while the ones from P2 are shown in cyan. The blue lines represent specular reflection paths.

8.3 Amortized Beam Tracing Results

We studied the trade-offs of the amortized beam tracing approach described in Section 6 by running a set of tests in which conservative beams were traced from source regions with varying sizes to find specular reflection paths between a source point (labeled 'A') and one of the three receiver points (labeled 'B,' 'C,' and 'D') in the 3D model shown in Figure 10.

During each test, cubes with width "Box Size" were constructed around both the source point and one of the receiver points. Then, the following three steps were used to compute specular reflection paths between the two points.

1. Trace conservative beams from both source regions up to 3 specular reflections.

2. Combine the beams to form polygon sequences containing potential paths with up to 5 specular reflections.

3. Process the polygon sequences to find all valid 5th-order specular reflection paths from the source point to the receiver.

Table 2 shows times (in seconds) measured during each test. The first column ("P") lists the receiver point, and the second ("Box Size") indicates the source box width in inches. The next four columns show the times required for beam tracing ("Beam Time"), polygon sequence construction ("Seq Time"), and path generation ("Path Time"), respectively. The column labeled "Amort Time" represents the average wall-clock time required for the amortized algorithm to compute all 5th-order specular reflection paths from the source to the receiver at even intervals separated by 1/2 inch, while the last column (labeled "Speedup") lists the speedup of the amorized beam tracing algorithm as compared to the bidirectional algorithm described in the previous section (i.e., "Box Size" = 0).

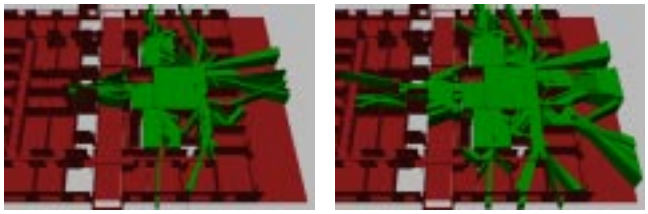
Scanning down the rows of the table, we see that the times required for all three steps increase with larger source regions, first slowly, and then very rapidly. This growth pattern matches the number of beams traced in each test. As the source region grows larger, a larger portion of the space can be reached by reflection paths, and

P	Box Size	Beam Time	Seq Time	Path Time	Amort Time	Speedup
B	0	4.2	0.48	0.004	4.691	1.0
	2	4.6	0.57	0.006	2.593	1.8
	4	5.1	0.64	0.006	1.423	3.3
	8	5.9	0.81	0.008	0.840	5.6
	16	8.9	1.48	0.013	0.651	7.2
	32	83.3	5.89	0.054	2.787	1.7
C	0	3.9	0.20	0.001	4.116	1.0
	2	4.3	0.23	0.001	2.264	1.8
	4	4.6	0.27	0.002	1.214	3.4
	8	5.4	0.39	0.005	0.730	5.6
	16	8.2	0.73	0.012	0.561	7.3
	32	13.9	3.07	0.049	0.531	7.8
D	0	2.5	0.12	0.000	2.597	1.0
	2	2.7	0.12	0.000	1.405	1.8
	4	2.8	0.14	0.001	0.739	3.5
	8	3.4	0.20	0.001	0.448	5.8
	16	5.2	0.35	0.004	0.344	7.5
	32	10.1	1.55	0.021	0.364	7.1

Table 2: Amortized beam tracing statistics.

more beams must be traced (see Figure 14). Accordingly, more polygon sequences are constructed, and they all must be processed for each path generation step. We must be careful not to choose source regions that are so big that the resulting beam tree and the list of polygon sequences are grossly over-approximating.

The benefits of conservative beam tracing are reaped by amortization as the beam trees and polygon sequences are re-used for more and more nearby avatar locations as the source regions grow. The result of this trade-off between extra processing versus amortization is shown in the right-most column of the table (labeled "Speedup"). In this experiment, we see that the competing factors are balanced most advantageously when the box sizes have width around 16 inches. In this case, the conservative beams do not overestimate exact ones by too much, and thus they require only 2.1x more time to compute, while the benefits of amortization can be realized over 16 distinct avatar locations separated by 1/2 inch as the avatar walks from the center of the box, resulting in an effective 7x speedup.



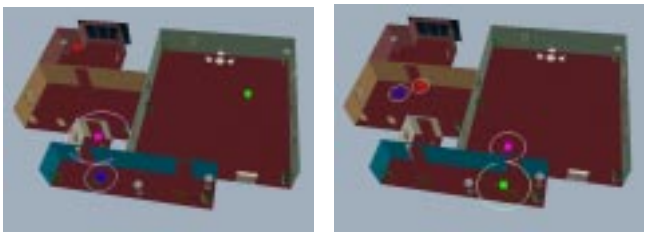
(a) Box Size = 0 (4,686 beams)

(c) Box Size = 32 (12,242 beams)

Figure 14: Beams traced for up to 2 reflections from source regions of different sizes centered at point 'A.'

8.4 Time-Critical Multiprocessing Results

We studied the time-critical multiprocessing aspects of our system by running tests with multiple avatars moving simultaneously through a virtual environment (shown in Figure 15) while logging statistics regarding the system's performance. During these tests, each of four avatars moved autonomously along a path which caused it to pass very close to other avatars sometimes, and to wander alone at others.



(a) T = 65 (b) T = 90

Figure 15: Visualization of adaptive real-time beam tracing (circles represent number of beam tree nodes).

Figure 16 contains a plot indicating the number of nodes in each of four avatars' beam trees during a test executing on an SGI Onyx2 workstation with 4 195 MHz R10000 processors. Each of the four curves represents a single avatar, and the vertical position of the curve at every time step indicates the amount of computation the system is expending updating the beam tree for the avatar. Figure 15 shows the status of the simulation at (a) T=65 and (b) T=95. For visualization purposes, the radius of the circle around each avatar in these images matches the corresponding height in the plot.

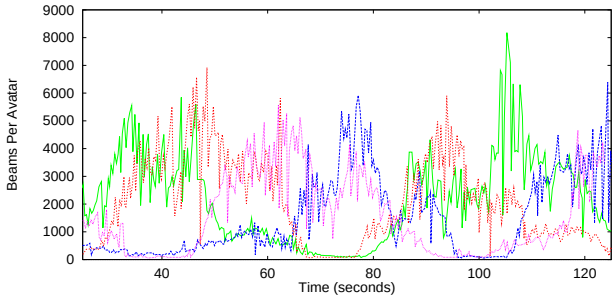


Figure 16: Plot of number of nodes in the beam tree for each avatar showing adaptive real-time beam tracing.

The plot clearly shows the adaptive and time-critical natures of our multiprocessing system. In some situations, such as the one occurring at T=65, two avatars are in close proximity, while the others are off by themselves. Accordingly, the system allocates most of its resources to computing beam trees for those two avatars (blue and purple) as they interact (i.e., the blue and purple lines are relatively high). At other times in the simulation, different avatars are interacting and the system adapts dynamically. Yet, the total number of beams traced for all avatars (i.e., the sum of the heights for all lines in the plot) remains approximately constant across all time steps. This feature reflects the time-critical nature of our real-time algorithm.

9 System 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. Nor do we trace beams along paths of refraction, diffraction, or diffuse reflection, which are very important acoustical effects. However, we think that the conservative beam tracing approach may be an appropriate framework for extending our system to handle these situations.

Second, our methods are only practical for coarse 3D models without highly faceted surfaces, such as the ones often found in acoustic modeling simulations. The difficulty is that beams get fragmented by cell boundaries as they are traced through a cell adjacency graph. For this reason, we are not optimistic that our implementation can be easily adapted to model light transport for photorealistic image synthesis. However, the algorithms probably can be applied effectively in other application areas concerned with simulation of wave phenomena (e.g., radio frequency propagation).

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.

Finally, our implementation does not include sophisticated models for acoustical reflectance distribution functions or for directionality of audio sources and receivers. For instance, we currently represent the reflectance of each surface with an angle-independent and frequency-independent absorption coefficient. However, since we compute reverberation paths explicitly, adding these auralization features is relatively straight-forward.

10 Conclusion

In this paper, we have described three beam tracing algorithms well-suited for geometric acoustic modeling in a distributed virtual environment system. These algorithms offer two important advantages over previous approaches: 1) they are much faster, and 2) they support real-time computing.

To summarize the speedup results, the priority-driven algorithm accelerates searches for early reflection paths by 2.6x – 4.3x; the bidirectional approach runs 4.8x – 6.2x faster than a comparable unidirectional algorithm; and, amortized beam tracing achieves speedups of 7.2x – 7.5x. Moreover, our multiprocessing system achieves nearly linear speedups for up to at least four processors. Overall, the speedup achieved combining all these algorithms is approximately two orders of magnitude over previous beam tracing systems.

Our adaptive multiprocessing system utilizes these new beam tracing algorithms to support many features essential for a real-time system, including time-critical updates, graceful degradation, and adaptive refinement. Moreover, the system performs asynchronous beam tracing computations, while reverberation paths are generated, and even extrapolated, at very high update rates. Overall, our system is able to match the accuracy of many previous off-line acoustic modeling systems while executing in real-time.

Acknowledgements

The authors thank Gary Elko, Mohan Sondhi, Jim West, and Perry Cook for their valuable discussions. We would also like to acknowledge Nadia Magnenat-Thalmann, Daniel Thalman, and Eric Petajan for use of the images in Figure 1.

References

[1] Allen, J.B., and D.A. Berkley, Image Method for Efficiently Simulating Small-Room Acoustics, *J. Acoust. Soc. Am.*, 65, 4, Apr 1979, 943-950.

[2] Arvo, James. Backward Ray Tracing. *Developments in Ray Tracing Course Notes*, SIGGRAPH 86, 1986.

[3] Begault, Durand, *3D Sound for Virtual Reality and Multimedia*, Academic Press, 1994.

[4] Blanchard, C., S. Gurgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, and M. Teitel, Reality Built for Two: A Virtual Reality Tool. *ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics*, (Snowbird, Utah), 1990, 35-36.

[5] Borish, Jeffrey. Extension of the Image Model to Arbitrary Polyhedra. *J. Acoust. Soc. Am.*, 75, 6, June, 1984, 1827-1836.

[6] Briere, Normand, and Pierre Poulin, Hierarchical View-Dependent Structures for Interactive Scene Manipulation, *Computer Graphics* (SIGGRAPH 96), 83-90.

[7] Chattopadhyay, Sudeb, and Akira Fujimoto, Bi-directional Ray Tracing, *Computer Graphics 1987 (Proceedings of CG International '87)*, Springer-Verlag, Tokyo, 1987, 335-343.

[8] Dadoun, N., D.G. Kirkpatrick, and J.P. Walsh. The Geometry of Beam Tracing. *Proceedings of the Symposium on Computational Geometry*, Baltimore, June, 1985, 55-61.

[9] Davison, B. *Neutron Transport Theory.* Oxford University Press, London, 1957.

[10] Drettakis, George, and Francois Sillion. Interactive Update of Global Illumination Using a Line-Space Hierarchy. *Computer Graphics* (SIGGRAPH 97), 1997, 57-64.

[11] Durlach, N.I., R.W. Pew, W.A. Aviles, P.A. DiZio, and D.L. Zeltzer. *Virtual Environment Technology for Training (VETT).* Report No. 7661, Bolt, Beranek, and Newmann, Cambridge, MA, 1992.

[12] Durlach, N.I, and A.S. Mavor, editors, *Virtual Reality Scientific and Technological Challenges*, National Research Council Report, National Academy Press, Washington, D.C., 1995.

[13] Foster, S.H., E.M. Wenzel, and R.M. Taylor. Real-time Synthesis of Complex Acoustic Environments. *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 1991.

[14] Funkhouser, Thomas A., Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West A Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments. *Computer Graphics* (SIGGRAPH '98), Orlando, FL, July, 1998, 21-32.

[15] Hart, P.E., N.J. Nilsson, and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on SSC*, Vol. 4, 1968.

[16] Hartman, Jed, Josie Werneck, *VRML 2.0 Handbook*, Addison-Wesley, ISBN 0-201-47944-3, August 1996.

[17] Hartmann, W.M., Listening in a Room and the Precedence Effect, *Binaural and Spatial Hearing in Real and Virtual Environments*, edited by Robert H. Gilkey and Timothy R. Anderson, Lawrence Erlbaum Associates, 1997.

[18] Heckbert, Paul, and Pat Hanrahan. Beam Tracing Polygonal Objects. *Computer Graphics* (SIGGRAPH 84), 18, 3, 119-127.

[19] Heckbert, Paul. Adaptive Radiosity Textures for Bidirectional Ray Tracing. *Computer Graphics* (SIGGRAPH 90), 24, 4, 145-154.

[20] Immel, David S., Michael F. Cohen, and Donald P. Greenberg. A Radiosity Method for Non-Diffuse Environments. *Computer Graphics* (SIGGRAPH 85), 19, 3, 133-142.

[21] Kleiner, Mendel, Bengt-Inge Dalenback, and Peter Svensson. Auralization – An Overview. *J. Audio Eng. Soc.*, 41, 11, Nov 1993, 861-875.

[22] Krockstadt, U.R. *Calculating the Acoustical Room Response by the Use of a Ray Tracing Technique*, J. Sound and Vibrations, 8, 18, 1968.

[23] Kuttruff, Heinrich *Room Acoustics*, 3rd Edition, Elsevier Science, London, England, 1991.

[24] Lafortune, E.P., and Y.D. Willems, Bi-directional path tracing, *CompuGraphics*, Alvor, Portugal, 1993, 145-153.

[25] Lewins, Jeffery. *Importance, The Adjoint Function: The Physical Basis of Variational and Perturbation Theory in Transport and Diffusion Problems.* Pergamon Press, New York, 1965.

[26] Moore, G.R. *An Approach to the Analysis of Sound in Auditoria.* Ph.D. Thesis, Cambridge, UK, 1984.

[27] *Quake*, id Software, Mesquite, TX, 1996.

[28] Smits, Brian, James R. Arvo, and David H. Salesin. An Importance-Driven Radiosity Algorithm. *Computer Graphics* (SIGGRAPH 92), 26, 2, 273-282.

[29] Sony Corporation, *Community Place Browser Manual*, 1996.

[30] Teller, Seth., and Carlo Séquin, Visibility Preprocessing for Interactive Walkthroughs, *Computer Graphics* (SIGGRAPH 91), 25, 4, 61-69.

[31] Teller, Seth Computing the Antiumbra Cast by an Area Light Source. *Computer Graphics* (SIGGRAPH 92), 26, 2, 139-148.

[32] Teller, Seth *Visibility Computations in Densely Occluded Polyhedral Environments.* Ph.D. thesis, Computer Science Division (EECS), University of California, Berkeley, 1992. Also available as UC Berkeley technical report UCB/CSD-92-708.

[33] Tsingos, Nicolas, and Jean-Dominique Gascuel. A General Model for Simulation of Room Acoustics Based On Hierarchical Radiosity. Technical Sketches, *SIGGRAPH 97 Visual Proceedings*, 1997.

[34] Veach, Eric, and Leonidas Guibas, Bidirectional Estimators for Light Transport, *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June, 1994, 147-162.

[35] Zyda, Michael J., David Pratt, John Falby, Chuck Lombardo, and Kristen Kelleher, The Software Required for the Computer Generation of Virtual Environments. *Presence*, 2, 2 (March 1993), 130-140.