Distributed Sound Rendering for Interactive Virtual Environments

Ken K.P. Chan 1
Rynson W.H. Lau 1,2
1 Department of CEIT, City University of Hong Kong, Hong Kong
2 Department of Computer Science, City University of Hong Kong, Hong Kong

Abstract
Sound rendering deals with adding environmental effects to localized sound sources. There are a few sound rendering methods proposed, with different targets of application, such as architectural simulation and modeling of multi-user interactive environments. To improve the rendering performance, some of these methods employ parallel hardware architectures or dedicated sound processing hardware. In this paper, we propose a distributed sound rendering architecture for real-time sound rendering in an interactive environment with moving observers and sound sources. We also propose the distributed prioritized sound rendering method to improve the rendering performance while preserving the perceptual quality. The new architecture has the advantages that it is based on low cost PCs and is scalable.

1. Introduction
In a collaborative virtual environment, geographically separated users may communicate and interact with each other through connected networks. Such communications and interactions may be via different media. Although existing work mainly focuses on visual communications, i.e., 3D graphics rendering [11], audio is beginning to receive more attention, due to its importance in human communication and the availability of consumer-level sound cards. A distributed system approach is developed for sound rendering [5], they are far from consumer-level usage due to their dependence on costly dedicated hardware. In contrast, our method is targeted for PCs and consumer-level sound cards. A distributed system approach is employed to share the sound rendering workload among the PCs. With our architecture, the sound rendering system is scalable according to adjustable performance requirements. As will be shown later, our initial experimental results are very encouraging. The main contributions of this work can be summarized as follows:

- We propose a scalable, distributed architecture for real-time sound rendering.
- We propose the distributed prioritized sound rendering method to support progressive real-time sound rendering.

The rest of the paper is organized as follows. Section 2 presents a brief survey on related work. Section 3 presents our distributed sound rendering architecture and the distributed prioritized sound rendering method. Section 4 shows and discusses some initial experimental results of the proposed methods. Finally, section 5 suggests possible future work.

2. Related Work
Sound rendering is the process of generating impulse responses to an observer from the input sound sources. These impulse responses may assist the observer to identify the direction of sound sources and acoustical characteristics of the environment. There are two approaches to sound rendering, physical approach and perceptual approach [1]. The physical approach deals with the geometrical modeling of sound via transmissions, reflections, refractions and diffractions. The impulse response is generated by convoluting sound waves with the corresponding HRTFs of virtual sound source positions, which are determined by the reachable sound rays from the sound sources. The perceptual approach, on the other hand, uses a parameterized model to approximate the acoustical characteristics of the environment. The impulse response is generated by adjusting a limited number of parameters such as room size and wall materials.

There are a few sound rendering methods proposed, which are based on the physical approach. For example, a beam tracing method was proposed in [2], which preprocesses all reflection paths. Accelerations of this method were presented in [9]. This method can achieve good results of discovering reflection paths once the beam tree is built. However, beam tracing has to deal with complex management of beams when intersecting beams with objects and fragmenting beams into smaller beams during occlusion. In addition, it cannot accelerate the rendering for environments with simple partitions but complex internal acoustic features.

A number of systems have implemented the physical approach for sound rendering. In [7], a scalable PC-based sound rendering system was presented, with the whole sound rendering process being implemented in software. However, the experimental results presented in the paper show that it cannot be used for real-time applications. There are also commercial hardware/software sound rendering systems [3,5]. [3] is a software system targeted for architectural acoustic simulation but it is not for interactive purpose. [5] is a hardware/software system for multi-purpose sound rendering. It makes use of dedicated DSPs to generate real-time spatialized sounds.
Although this system claims to be for interactive applications, it is not scalable and is very expensive.

Since the physical approach deals with recursive searching of sound transmissions and reflections, its complexity is much higher than the perceptual approach, which only uses a limited set of parameters to generate the impulse response. The perceptual approach has a constant complexity once we have defined impulse response parameters for each spatial subdivision. For the physical approach, we have to traverse logarithmic number of surfaces for each ray projection. After each intersection, two new rays are generated: transmission ray and reflection ray. However, for deeper level searches, the sound wave will have passed through many surfaces and hence its intensity will be significantly attenuated. Hence, we can set a threshold to limit the search level to reduce the computational cost.

As sound wave is both transmissive and specularly reflective, the physical approach has borrowed many ideas from the ray-tracing methods used for image rendering. Acceleration techniques developed for ray-tracing can be generalized into three approaches: pre-processing data structures, caching by exploiting ray coherence and distributed ray-tracing. Pre-processing data structures include grids, BSPs and BVHs [6]. By applying these data structures, the number of intersection tests for each ray can be reduced. For exploiting ray coherence, caching of ray paths may be considered as rays start at the same point and travel in similar direction will likely intersect the same object. Tracking of ray paths across frames may also be used to exploit frame coherence [8].

In our framework, the physical approach is employed for the reason that it can generate more accurate early reflection cues, which are important for the observer to sense the environment. The validity of the physical approach has been investigated in [10] and results have shown good matches between simulated and measured impulse responses.

3. Distributed Sound Rendering

3.1 System Architecture

In a multi-user virtual environment, clients may communicate with each other through real-time speech. Our sound rendering architecture, as shown in figure 1, supports not only real-time transmission of speech among multiple users, but also environment modeling. There are three kinds of servers in the system: the sound rendering servers, the auralization servers and the coordination server. The sound rendering servers receive sound rendering requests from the coordination server and search for virtual sources by ray-tracing. An impulse response, which represents the directions and gains of virtual sources, is then generated and sent to the auralization servers. The auralization servers convolve the received impulse response with sound source waveforms to auralize the spatial sound for each client. The coordination server acts as the interface between clients and the sound rendering servers. Ray-tracing jobs are dispatched by the coordination server to the sound rendering servers.

The sound rendering servers share the workload of sound rendering, while the auralization servers share the workload of auralization. The coordination server facilitates load balancing among these servers. The distribution of ray-tracing work follows a mixed data/demand driven approach, as will be discussed next.

3.2 Distributed Prioritized Sound Rendering

The distributed prioritized sound rendering method is developed for time-critical applications while maintaining a maximum level of spacious quality by means of job prioritization and distributed processing. Its aim is to generate a progressively refined set of virtual sound sources in a time-critical manner. As ray-tracing is an expensive process, we prioritize the rays to select the perceptually more important rays for projection to optimize the processing time. The prioritization can be done by searching reverberation paths in a breadth-first order rather than the traditional depth-first order.

During the period between two motion samples of a moving observer, the rendering server would continuously pass progressively refined impulse responses to the auralization server. Rather than waiting for the whole ray-tracing process to be completed before the observer can hear the rendered sound (Figure 2), the observer can hear the partial results once any reverberation paths are generated (Figure 3). When further reverberation paths are generated, the later results would be added to the impulse response so that the final sound would be more and more spacious.
In most sound rendering methods, such as [4], interpolation is usually used to smooth the discontinuity of spacious sound between two impulse response generations. Interpolation, in general, would produce least distortion if the frame period is short and the change between two consecutive frames is small. Our method progressively refines the impulse responses through short time intervals, reducing the distortion caused by interpolation. It also ensures a more constant frame rate and hence a fast moving observer can still hear coarsely rendered sounds along its path. Although the spaciousness is coarser during a fast motion, it is perceptually much better for the observer than to hear discontinuous sound at an unsteady low frame rate.

### 3.2.1 Job Prioritization

Our job prioritization approach is similar to the priority-driven method proposed in [9], which prioritizes the beams for beam tracing. However, instead of applying the approach in the preprocessing stage, we apply it during the run-time ray projection stage. A best-first search is applied to search for virtual sound sources by their importance. The major computational cost of ray-tracing is on ray intersections. Instead of a depth-first search, we use a heuristic search strategy to choose the most important rays for projection and eliminate unnecessary ray intersections. The importance criterion is a run-time heuristic function. A useful criterion is the gain of a sound source to the observer.

For an arbitrary intersection point $i$, we estimate its direct distance to a potential virtual source $s$ as $d_{i,s}$. The estimated gain $g_{i,s}$ of $s$ perceived by the observer can be determined as:

$$g_{i,s} = \frac{D_{i,s} \times G_i \times \sum_{j=0}^{i-1} (r_j \times t_j) \times D_{i-1,j}}{d_{i-1,j}}$$

where $f_i$ is the attenuation factor accumulated from previous intersection points.

We set a threshold gain $G_{\text{min}}$ to represent the minimum audible gain of the observer. At an intersection point, if all $g_{i,s}$ for all sound sources are smaller than $G_{\text{min}}$, we can terminate the search since we will not be able to hear them anyway. Note that since $d_{i,s}$ is an underestimate, we can guarantee that the actual ray distance between $i$ and $s$ must be longer than $d_{i,s}$.

For each reverberation path $p$, where $1 \leq p \leq P$ and $P$ is the total number of reverberation paths, we compute a gain $g_p$, which is estimated as the maximum value of all $g_{i,s}$ at the ray intersection point, i.e., the maximum gain among all potential virtual sound sources. Whenever we want to further project a ray, we simply choose the one with highest $g_p$ for deeper level search. This approach basically prioritizes the search so that virtual sources with higher gain are processed first.

### 3.2.2 Distributed Processing

When we put the job prioritization approach in a distributed processing environment, we use a mixed data/demand driven approach for job dispatching. The comparison among $g_p$’s implies that a synchronization mechanism is needed. A demand driven approach is thus required. However, simple synchronization may cause some servers to be idled. Our design hence allows certain degree of independence among servers, taking the advantage of data driven approach.

Figure 4 illustrates our method with an example. First we dispatch the primary rays of all reverberation paths to all servers in a round-robin manner. In the example, reverberation paths $\{p_1, p_5\}$ are dispatched among 3 rendering servers. Each server would return its $g_p$ to the coordination server for prioritization. At checkpoint 1 all the initial $g_p$’s are collected and prioritized. We use a window to limit the number of paths for further tracing. For a window size of 3, we only trace the first 3 dominant paths in the next pass. Hence after checkpoint 1, only paths $p_2$, $p_5$, and $p_6$ are traced. To avoid the servers from idling, we allow the servers to further trace their own paths until new assignments are received from the coordination server. In the example, jobs $p_2$ and $p_5$ finish before checkpoint 1. They are further traced as long as there are unfinished jobs in other servers. At checkpoint 2, we collect new values of $g_{p_2}$, $g_{p_5}$, and $g_{p_6}$, and carry on to next prioritization.

The window size could be adjusted to meet with different requirements. A small window size produces the initial impulse response faster. However, there is a possibility that paths with initially low estimated gains are actually dominant paths. A large window size thus facilitates a fairer prioritization by allowing development opportunity of weaker paths.

Data coherency and efficiency are considered in job distribution. In our example, at checkpoint 2, $p_2$ is assigned to server 1 as server 1 has the latest trace record of $p_2$. This facilitates data coherency. On the other hand, at checkpoint 1, $p_6$ is assigned to server 1 as server 3 is busy on $p_3$. Here, we choose to transfer $p_6$ to another server for efficiency.
3.3 Scalability of the System

By applying the above method in our distributed architecture, the system can be scalable according to performance requirement. To maintain a desired sound rendering frame rate of a complex scene, we may choose to add more rendering servers, to deepen fewer reverberation paths by limiting the window size, or to increase the value of the gain threshold $G_{\text{min}}$. If we add more servers, the round-robin period would be shorter and the general response time would be shorter. Alternatively, we may decrease the window size so that fewer paths would be chosen for further tracing. This will guarantee that important reverberation paths could be found within a time limit although fewer paths are generated. In this case, spaciousness of sound has been traded for efficiency. However, as will be shown in our experimental results, the error is minimized. Another alternative is to increase $G_{\text{min}}$ but as we cannot determine the cut-off gain (the gain obtained when the future rays will not be projected) each time, $G_{\text{min}}$ may be overestimated or underestimated. Thus this option is undesirable.

4. Experimental Results

We have implemented the proposed architecture on a set of Intel Pentium 4 2GHz machines, connected by a 100Mbps local area network. In our experiment, we study the perceptual characteristics of our sound rendering prototype system. We measure the perceived gain against the number of rays projected.

Figure 5 shows the gain development curve during the ray-tracing process. Prioritization is done after each pass of ray projections where the number of ray projections equals to the window size. A smaller window size implies more tracing passes and thus more prioritization operations. The total number of reverberation paths to search is 400. In the experiment, the first 400 ray projections are common to all window sizes. The difference is shown after the 400th projection. For a window size of 30, the dominant reverberation paths are found quickly during the initial passes. For a window size of 200, it takes longer to find the dominant reverberation paths. For a window size of 400, which has the same effect as traditional ray-tracing, it takes even longer to find the dominant reverberation paths and abrupt changes appear in the gain curve.

Hence, a smaller window size implies that we can obtain the dominant reverberation paths at early stage of rendering and the output sound quality can be progressively refined. In addition, a point of saturation can be detected from this curve so that we can choose to stop the rendering process earlier. In figure 5, with a window size of 30, we can still get the dominant reverberation paths if we stop rendering after 500 projections. In contrast, the gain curve of the traditional ray-tracing method may change suddenly in later stages, which is much less desirable in a progressive rendering environment.

We have also investigated the response times of our prototype versus different numbers of rendering servers used. We observed that the response time drops sub-linearly with increasing number of servers. A high number of servers introduces higher network connection and job exchange overheads among the servers and hence offsets the advantage of adding additional servers.

5. Future Work

The initial feedbacks from the users experimenting with our prototype system are very encouraging. We are currently investigating techniques to apply our method on large scenes with large number of users. In such situation, scene partitioning and a distributed scheme for job coordination should be considered for efficient rendering. We would also like to validate our methods using real-world models. The perceptual impact on early rendering cut-offs would be investigated.

6. References