Progressive Sound Rendering in Multimedia Applications

Qiong ZHANG  Jiaoying SHI
State Key Lab of CAD&CG, Zhejiang University

ABSTRACT
Realistic audio rendering is an integral part of any immersive multimedia applications. A progressive sound-rendering scheme, which integrates sound source modeling with sound transmission simulation, is proposed to generate 3D sound effects for multimedia applications. At first, sound source modeling is performed through either sinusoidal analysis of prerecorded sound or modal analysis of virtual object based on its geometry and material properties. Then, all the sub components derived are sorted according to perceptual criteria. Dependent on resource availability, the afterwards real time synthesis can be done in a progressive way and binaural filtering is integrated into this process directly.

1. INTRODUCTION
Sound has long been acknowledged as an effective channel in human-computer interaction [1]. Strictly speaking, there’re two primary steps involved in rendering realistic sound in multimedia and VR applications, i.e., sound source modeling and sound transmission modeling from sound source to the eardrums of the user [2][3]. Previous research on sound rendering largely focused on pure sound transmission modeling [4][5]. All of those efforts didn’t touch sound source modeling and prerecorded or pre-synthesized sounds are used instead.

Sound source modeling in interactive applications only became research target until very recently [7][8][9], van den Doel et al [7] developed a system for automatic generation sounds made by the real time contact interactions between solid objects. This system is based on a good physically motivated model called modal synthesis. The effectiveness of this model is further demonstrated by Avanzini et al [8] and O’Brien et al [9]. However, modal synthesis may not applicable to low-end platforms due to large amount of modes involved in the synthesis process. Furthermore, geometry data or material properties for an object are not always available. A complementary method is necessary to rendering sound in this case.

Figure 1. Flowchart of progressive sound rendering

In this paper, we propose a progressive sound-rendering scheme. This scheme can be divided into three consecutive steps as illustrated in Figure 1, i.e. sound source modeling, components sorting, and progressive synthesis.

2. SOUND SOURCE MODELING
Two methods are used in sound source modeling. Modal analysis is selected if modeling contact sound and geometry and material properties are available. Otherwise, sinusoidal analysis will be used instead.

2.1 Modal Analysis for Contact Sound
Modal analysis and synthesis is based on following equation [7][9]:

\[ Q(t) = \sum_{i=1}^{n} A_i e^{-d_i t} \sin \omega_i^d t \]  

Where \( A_i \) is the mode gain, \( d_i \) is the decay rate, and \( \omega_i^d \) is the mode frequency.

Assuming a discretized object has \( L \) nodes on the surface and \( 3l-2, 3l -1, 3l \) are three corresponding degree of freedom at node \( l \). The deviation from equilibrium of the surface at this node under impulse force could be described by:

\[ q_i(t) = (y_{3l-2}(t) + y_{3l-1}(t) + y_{3l}(t))n_i \]

\[ = \sum_{l=1}^{n} (a_{i(3l-2)}n_x + a_{i(3l-1)}n_y + a_{i(3l)}n_z)e^{-d_i t} \sin \omega_i^d t \]

Where \( n_i \) is the normal vector at node \( l \), \( n_x, n_y \) and

0-7803-8603-5/04/$20.00 ©2004 IEEE.
\( n_z \) is the unit vector in X, Y and Z direction respectively, and \( y_{3L-2}(t) \), \( y_{3L-1}(t) \) and \( y_{3L}(t) \) are the node displacements. Considering every node on the surface could be seen as an individual sound source, the sound actually emitted from the vibrating solid object is very complicated and could be roughly expressed as the summation of the surface deviation of all the surface nodes:

\[
Q(t) = \sum_{i=1}^{L} q_i(t) = \sum_{i=1}^{n} A_i e^{-d_i t} \sin \omega_i^d t
\]  

Where \( A_i = \sum_{l=1}^{L} (a_{i(3l-3)} n_{i, n_{l}} + a_{i(3l-2)} n_{i, n_{l}} + a_{i(3l-1)} n_{i, n_{l}})
\)

Among modal synthesis parameters, both decay rate \( d_i \) and mode frequency \( \omega_i^d \) are independent of simulation context and could be fully derived from the geometry and material properties of the object \([7][9]\). Only mode gain \( A_i \) needs to be resolved in real time.

### 2.2 Sinusoidal Analysis for Other Sound

The basic idea behind sinusoidal analysis algorithm is quite simple \([6]\); audio signal is transformed from time domain to frequency domain with short-time FFT, frame by frame. Then salient frequency-domain parameters are extracted to represent original audio signal. Synthesis is performed based on those extracted parameters.

Basically, the parameters extracted at each frame include peak frequencies \( f_i^l \) and their corresponding magnitudes \( M_i^k \) and phases \( \theta_i^k \), \( l \in [1, L'], k \in [1, N] \).

Where \( L' \) is the number of peak frequencies at frame \( k \) and \( N \) is the frame number in total.

In order to synthesize smooth-transition audio signal in time domain, interpolation is usually used to fade in/out audio parameters for consecutive frames. In order to do that, peak matching process is first executed to establish matching pairs \((M_i^k, f_i^k \theta_i^k)\) and \((M_i^{k+1}, f_i^{k+1} \theta_i^{k+1})\) between adjacent frames. Then interpolation is performed to attain magnitude \( M_i(t) \) and instantaneous phase \( \theta_i(t) \) for each matching pair. The final synthesis is based on following equation:

\[
\hat{x}(t) = \sum_{l=1}^{L} M_i(t) \cos \theta_i(t)
\]  

### 3. COMPONENTS SORTING

As we can see, both modal analysis and sinusoidal analysis will result in large amount of audio components, i.e. mode frequencies in modal analysis and matching pairs in sinusoidal analysis. For networked or real time multimedia applications, it can be very beneficial if we can sort those components based on perceptual criteria so that final sound can be synthesized progressively. Here frequency-masking effects \([11]\) is employed in sorting components.

#### 3.1 Mode Frequencies Sorting

Unlike progressive geometry models that we could build in advance, one of the modal synthesis parameters, i.e., mode gain is dependent on the interaction context. It makes sorting of mode frequencies more challenging.

Van den Doel et al \([10]\) proposed a method to order modes. However, it only works for the contact force applied at predefined positions and directions since mode gains used are predetermined rather than calculated in real time.

Obviously, mode gain plays a very important role in measuring the perceptual significance of each mode. It’s desirable to have modes ordered in real time. In order to achieve this, a two-step procedure including offline preprocessing and subsequent real time calculation is adopted.

During preprocessing, all the mode frequencies \( \omega_i^d \) in audible range are transformed into Bark scale \([11]\) \( BARK_i \). Then, following mode gain sampling process is executed: assuming a unit impulse force is applied at node \( l \) at time 0 in X direction, calculate mode gain \( A_i^{lx} \).

Similar, we could calculate \( A_i^{ly} \) and \( A_i^{lz} \) respectively.

By applying the unit impulse force at different nodes \((l=1, 2, 3, \ldots, L)\) and in different directions (X, Y and Z), the mode gains are sampled 3L times in total. Due to the linearity of modal synthesis, mode gain under any impulse force could be expressed in linear combination of those sampled mode gains.

In the real time simulation, mode gain \( A_i \) is calculated as the linear combination of sampled mode gains dependent on the applied location and direction of current contact force.

Similar with Doel’s method \([10]\), inter-modes masking effects is approximated by considering masking of narrowband noises at \( BARK_i \) with power \( A_i^2 / d_i \). All
the modes in a same critical band are summed together to obtain a single noise masker for each critical band like MPEG psychoacoustic model [1].

At first, any mode whose power is below the absolute threshold is discarded. Then, the additive masking effects of modes in different critical bands is modeled linearly and individual masking thresholds from or all the modes are combined to form a global masking threshold.

Finally, all the modes are ordered by the difference between the mode power and masking threshold at this frequency. By that way, all the modes can be sorted dynamically in the real time simulation. The number of modes eventually used in final sound synthesis is dependent on the availability of system resources.

### 3.2 Matching Pairs Sorting

After getting all the matching pairs in sinusoidal analysis, we will sort them according to the criteria similar with MPEG psychoacoustic model 1. At first, the power density spectrum at each peak frequency for each frame is calculated:

\[ PDS_i^k = 10 \times \log_{10} |M_i^k|^2 \text{ dB}, \ i \in [1, L^i], k \in [1, N] \]

Where \( L^k \) is the number of matching pairs at frame \( k \) and \( N \) is the frame number. Then, global masking threshold \( GMT_i^k(i) \) at each frame is calculated. Later on, any matching pair whose both magnitudes are below global masking threshold are discarded. Finally, according to relative distance between \( \{ PDS_i^k, PDS_i^{k+1} \} \) and corresponding \( \{ GMT_i^k(i), GMT_i^{k+1}(i) \} \), we will reorder all the matching pairs for every consecutive frame pair. That is, the most salient matching pairs will be placed at the beginning of matching pair sequence and the lease salient one be the end of this sequence.

### 4. PROGRESSIVE SOUND SYNTHESIS

As we know, sound transmission modeling usually includes two parts, namely environment and pinnae filtering. Environment filtering is used to model environmental context. In psychoacoustics, pinnae filtering is represented by Head-Related Impulse Response (HRIR) in time domain or Head-Related Transfer Function (HRTF) in frequency domain. HRTF is a function of three parameters, i.e. azimuth, elevation, and frequency. The convolution of mono sound signal with appropriate HRIR is typically featured as the key component of realistic sound generation in virtual environment. However, the computations involved in convolving the sound signal from a particular point in space are quite huge. Usually, it couldn’t be done in real-time without special DSP hardware.

The equivalent process for convolution in frequency domain is multiplication. In modal synthesis, since every mode can be seen as a narrowband noise or pure tone when decay is very weak, those modes could be roughly thought as the spectral representation of an audio signal. Thus, HRTF filtering can be applied to those individual modes first before actual synthesis. This will decease computational complexity substantially by avoiding time-domain convolution.

Similarly, HRTF filtering could be naturally integrated into sinusoidal synthesis. The reason is that we have already extracted salient frequency-domain parameters, i.e., magnitude and phase information at each peak frequency. It’s natural to consider the synthesis process and the generation of 3D realistic audio at the same time.

The actual synthesis procedure is as follows:

Step 1: Assuming the location of sound source relative to user is at azimuth = \( \theta \) and elevation = \( \varphi \), determine corresponding HRTF\(_{\theta, \varphi}\).

Step 2: Modal Synthesis -- For every mode frequency \( \omega_i^d \), find corresponding HRTF\(_{\theta, \varphi}(\omega_i^d)\) at the closest frequency point.

Sinusoidal Synthesis -- Determine corresponding HRTF\(_{\theta, \varphi}(f_i^k)\) and HRTF\(_{\theta, \varphi}(f_i^{k+1})\) for peak frequency \( f_i^k \) and \( f_i^{k+1} \) from two adjacent frames.

Step 3: Modal synthesis -- Modal synthesis equation

\[ A_i e^{-wd_i} \sin(\omega_i^d t) \text{ transformed into: } Q_i(t) = MAG[HRTF_{\theta, \varphi}(\omega_i^d)]A_i e^{-wd_i} \]

\[ \ast \sin(\omega_i^d t + \text{PHASE}[HRTF_{\theta, \varphi}(\omega_i^d)]) \]

Where MAG and PHASE is the magnitude and phase of HRTF respectively.

Sinusoidal Synthesis -- Calculate new amplitude/phase \( M_i^k \), \( \theta_i^k \), \( M_i^{k+1} \), and \( \theta_i^{k+1} \) at peak frequency \( f_i^k \) and \( f_i^{k+1} \) after considering HRTF filtering:

\[ \{ M_i^k, M_i^{k+1} \} = MAG[HRTF_{\theta, \varphi}(f_i^k)] \ast \{ M_i^k, M_i^{k+1} \} \]

\[ \{ \theta_i^k, \theta_i^{k+1} \} = \text{PHASE}[HRTF_{\theta, \varphi}(f_i^k)] + \{ \theta_i^k, \theta_i^{k+1} \} \]

Step 4: The final modal synthesis is based on following equation:

\[ Q_i(t) = \sum_{i=1}^{n} Q_i(t) \]

The final sinusoidal synthesis is based on equation (4).
The progressive sound-rendering scheme proposed above is very suitable for distributed implementation. Figure 2 illustrates the basic architecture of a possible solution when it is used in networked multimedia applications.

Figure 2. Application of Progressive Sound Rendering in Networked Multimedia Applications

At server side, the Collector is responsible for preparing matching pair pool and mode frequency pool. The Distributor is responsible for sending out matching pairs or mode frequencies based on client feedback. At client side, there's a HRTF data pool for each user. Different user can choose different set of HRTF data. The Synthesizer is responsible for resynthesizing audio signal and sending feedback to the server side.

By this way, we can adjust the output quantity of matching pairs/mode frequencies based on feedback from client/network. That is to say, if the computational power at client side can't fulfill the synthesis requirement, the client will ask server to decrease the number of matching pairs/mode frequencies for following audio frames. By that way, different clients may receive different number of matching pairs that can result in different hearing quality. Furthermore, since each user can choose his personal HRTF data set for 3D audio rendering. It provides a possible way to use individual HRTF rather than collective HRTF, which can further guarantee the synthesis quality.

5. CONCLUSION

A progressive sound-rendering scheme is introduced. It includes three consecutive steps, i.e., sound source modeling, components sorting and progressive synthesis. Although sound source modeling may lead to a few hours of computation, this preprocessing step could be done offline and have no impact on real time simulation. Preliminary results validate the correctness and effectiveness of our scheme. Further research will be focusing on better binaural model and room acoustic simulation integration.

REFERENCES