ACTA ACUSTICA UNITED WITH ACUSTICA
Vol. 95 (2009) 106 – 117
DOI 10.3813/AAA.918132

Frequency Domain Acoustic Radiance Transfer for Real-Time Auralization

Samuel Siltanen, Tapio Lokki, Lauri Savioja
Department of Media Technology, Helsinki University of Technology, P.O.Box 5400, 02015 TKK, Finland.
Samuel.Siltanen@tml.hut.fi

Summary
A method for modeling room acoustics and auralizing the results in real time with a moving listener is presented. The acoustics modeling is based on the acoustic radiance transfer technique which is capable of modeling arbitrary reflection properties of different materials. The novel idea of implementing this technique in frequency domain allows modeling of all frequencies at once as the time domain technique requires separate runs for each frequency band. Since the auralization of the results requires scaling, adding, and delaying responses as well as convolving them with head-related impulse responses, the massive parallel computation capacity of modern graphics hardware is utilized. Thus, realistic interactive walkthroughs are possible in typical room models with a stationary source.

PACS no. 43.55.Ka, 43.58.Ta

1. Introduction

Interactive simulation of sound propagation in a complex space is important for creating realistic virtual environments. The methods used for room acoustics modeling today are either offline algorithms, use simplified reflection models, or enable realtime walkthrough using a perceptual model without a rigid physical basis. In this paper, a technique for interactive physically-based simulation of sound propagation in virtual environments with arbitrary reflection properties is presented. Figure 1 illustrates this technique.

The proposed algorithm is based on a previously introduced acoustic radiance transfer method which produces results that are validated against real-life measured data [1]. The output of this method is a complete room impulse response, and not only the early part containing low-order specular reflections. The high performance is achived by utilizing the fact that the time-consuming part of the method can be pre-computed and only the last phase of the method must be run in realtime when the listener moves.

Previously, the acoustic radiance transfer method has been implemented in the time domain and responses for different frequencies had to be modelled separately. The new method works in the frequency domain and allows the responses for the full spectrum to be modelled at once. This simplifies the use of complex frequency-dependent reflection models.

Received 12 June 2008, revised 23 September 2008,
accepted 29 September 2008.

The trend in recent years has been that the pure computing power of graphics hardware has increased at a faster pace than that of central processing units. This massive parallel computing hardware has not yet been widely used for auralization purposes, although the nature of the problem is ideal for parallellism. Thus, the presented implementation utilizes the graphics hardware for processing the responses during the realtime computation.

The main contributions of the paper are the following:

- Realtime auralization in rooms with arbitrary reflection properties for a moving listener.
- Frequency domain computation of the acoustic radiance transfer.
- Utilization of the massive parallellism available on graphics hardware for realtime auralization.

The presented system enables dynamic realtime auralization of, e.g., a concert hall model. Such a tool would be of great help for acoustic consultants. They can give their clients a free walk in the virtual concert hall, and in addition to an impressive visualization, the clients can listen to the designed acoustics in any place of a virtual model.

The most relevant related work is briefly reviewed in the next section. Then, the concept of acoustic radiance transfer is introduced and it is shown how it can be utilized in a method for modeling room acoustics. It is shown that the operators required in this method can also be applied in the frequency domain, and the fourth section covers the design of the novel implementation of the acoustic radiance transfer algorithm. This description includes a brief explanation of the pre-computation part and a more detailed discussion of the runtime computation and especially the effective use of graphics hardware. Eventually, the algorithm is run with four test models and the results are given, the limitations

© S. Hirzel Verlag · EAA

Figure 1. The acoustics of a concert hall model is simulated with the acoustic radiance transfer method. The results from all the surface patches are convolved in realtime with an anechoic stimulus to create a binaural interactive auralization.

of the method are listed and the possible future improvements are suggested.

2. Related Work

This section briefly reviews works that are related to realtime auralization, head-related transfer functions, and utilization of graphics procesing unit (GPU) in handling audio signals. Covering the numerous methods for computational acoustics modeling is out of the scope of this paper (an interested reader may consult the survey by [2]). Only the work most directly related to the presented work is referred to.

In **sound rendering** [3] the acoustic modeling results are made audible, i.e. auralized [4] by convolving the room impulse response with a dry audio signal. A straightforward method is to apply the impulse responses directly in convolution, but this approach is problematic when the listener moves and the impulse response varies over the space. For dynamic rendering it is more convenient to use the parametric room impulse response rendering technique in which each reflection path is processed separately [5, 6]. In that case only a limited number of early reflections can be processed, and an artificial reverberation algorithm has to be used to fill in the remaining part of the response. However, the novel approach of the presented work is to model the complete impulse response even for a moving listener.

Head-related transfer functions model the effects caused by the listener's head, shoulders, and pinnae, and how they modify the sound signals coming from different directions [7]. These functions are needed when making spatialized sound for headphones. In practice, auralization can be performed by convolving a dry signal with a head-related impulse response (HRIR) filters corresponding to the incoming direction of the sound, thus modeling the signal arriving at the eardrums of the listener. An alternative approach would be to use finite impulse response (FIR) or infinite impulse response (IIR) filters [8]. In the presented work, the FIR filtering approach is chosen for the direct sound, since it has to be processed in the time domain, for the reasons explained later, and the convolution would be too costly.

On the other hand, the reflections which are handled in the frequency domain can more easily utilize the convolution approach. When measured data is used, a discrete sampling of directions around the listener results in hundreds of HRTF filters for both ears to cover the whole space sufficiently well. Utilizing this kind of data, e.g. with linear interpolation, is impractical when the amount of memory is limited, and thus reduced models of the HRTF have been suggested.

There are techniques based on relatively simple models which try to emulate the characteristics of human pinna [9, 10]. The employed models, however, are quite crude approximations. Another technique that has been suggested models the external ear as a multisensor broandband beamformer [11], but the computational burden and the numerical instability become difficult issues when modeling wide solid angles.

One way to save memory is to compress a measured HRTF database by means of principal component analysis (PCA) [12, 13, 14]. The interaural time delay is separated and a minimum phase HRTF is represented as a linear combination of the principal components. A relatively small number of components are required to suppress the error to a reasonable level. Due to the minimum phase assumption the measured phase behaviour is lost.

Chen et al. propose using complex valued HRTFs which are expressed as a combination of a set of eigentransfer functions (EFs) [15]. The EFs are orthogonal set of frequency-dependent functions and they are obtained by a discrete Karhunen-Loève expansion procedure. The number of EFs required to produce good quality HRTFs can be even smaller than the number of PCA components, and the measured phase is also taken into account. The HRTFs are also continuous unlike in some cases in the PCA technique.

Yet another similar compression technique is based on spherical harmonics (SH) [16]. The produced HRTFs are continuous. The SHs are hierarchical and they have a known structure so that adding more coeffients increases the accuracy in a predictable way. However, more SH components and coefficients are needed to accurately represent the HRTF, because the components are not adapted to the specific HRTF modelled, unlike in the PCA components or EFs.

Since the decomposition into components saves memory, it is suitable for the presented work which does computation on the GPU, where the memory is limited. In addition the reconstruction of the HRTFs on the GPU does not significantly increase the computational cost as long

as the number of components is low. In the SH decomposition, the number of components required is unnecessarily high. The PCA approach was chosen for its simplicity and robustness, although the technique based on EFs could have produced better quality.

Using the GPUs in acoustic computations is a step towards interactive acoustics modeling. Gallo and Tsingos explained how to implement two typical signal processing tasks on the GPU: variable delay-line and simple filtering. They compare the performance of the GPU implementation to a CPU implementation, concluding that GPU could accelerate audio rendering by 20% back in 2004 and that the latest graphics architectures would improve the GPU performance significantly [17]. Recently, Tsingos et al. showed how to utilize the GPU in computing the first order scattering effects modeled with the Helmholtz-Kirchhoff integral theorem and the Kirchhoff approximation [18].

Röber et al. used an energy-based formulation of the sound propagation and utilized a GPU-based ray tracer to compute the reflection paths [19], which were then rendered audible. Their work is similar to the presented work, but instead of the ray tracing technique and frequency-band-based approach, the frequency domain radiance transfer technique is used, which allows general reflection models, much longer responses, and more accurate frequency-dependent effects.

The **acoustic radiance transfer** is a recently presented acoustic modeling technique based on progressive radiosity and arbitrary reflection models [1]. The acoustic energy is shot from the sound source to the surfaces of the model which have been divided into patches. Then, the propagation of the energy is followed from patch to patch and the intermediate results are stored on the patches. Finally, when the desired accuracy is achieved, the energy is collected from the patches to the listener. The details of the technique are discussed later in this paper. However, the original method is only an off-line algorithm. In this paper, the algorithm is modified to perform the acoustic radiance transfer in the frequency domain. This approach, in addition to the usage of the GPUs, makes it possible to run the final gathering and sound rendering in realtime thus enabling interactive walkthroughs in environments with arbitrary reflection properties.

3. Acoustic Energy Transfer

The acoustic radiance propagation in an enclosure can be written in the form of the room acoustic rendering equation:

$$\varrho(x', \Omega) = \varrho_0(x', \Omega) + \int_{\mathcal{G}} R(x, x', \Omega)\, \varrho(x, \Gamma)\, \mathrm{d}x, \qquad (1)$$

where the outgoing time-dependent radiance, $\varrho(x', \Omega)$, from a surface point x' in direction Ω is written as a sum of two terms: the radiance emitted by the surface itself, $\varrho_0(x', \Omega)$, and the radiance from all the other surfaces reflected via the surface point x. The latter term is an integral

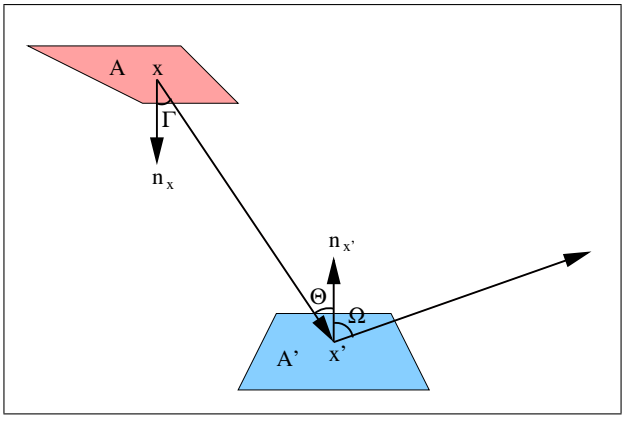

Figure 2. A single reflection from point x via point x' in direction Ω. Γ is the direction of the radiance leaving point x and Θ is the direction is which the radiance arrives at point x'. The directions are two-dimensional quantities which can be expressed, e.g. with elevation and azimuth angles. The surface normals are n_x at point x and $n_{x'}$ at point x'.

over $\mathcal{G} \subset \mathbb{R}^3$ which is the set of all surface points in the enclosure. The radiance from one of these points, x, towards point x' is denoted as $\varrho(x, \Gamma)$, where Γ is the direction of the radiance. Since only a fraction of the radiance leaving point x actually reaches point x', the radiance is multiplied by

$$R(x, x', \Omega) = \mathcal{V}(x, x')\, \rho(x', \Theta, \Omega)\, g(x, x'), \qquad (2)$$

where $\mathcal{V}(x, x')$ accounts for the occlusion, $\rho(x', \Theta, \Omega)$ is the bi-directional reflectance distribution funtion (BRDF) which describes the portion of the radiance arriving at point x' from direction Θ reflected at direction Ω, and $g(x, x')$ is a function taking into account the attenuation by distance, the angles between the surfaces, and the propagation delay, since the speed of sound is finite and this causes temporal spreading of the radiance. These variables are described in detail in [1], but for the following discussion it is sufficient to note that this kind of relationship can be written for the transfer of radiance between two surface points. Figure 2 shows the relevant geometry in the case of a single reflection.

A similar relationship can be written between two convex surface areas, i.e. patches. If the patches are small enough compared to their distance, the propagation delay and directional properties of the radiance from one patch to the other patch remain approximately constant over all points on the patches. Then the fraction of the radiance leaving patch A and arriving at patch A' is called the form factor, $F(A, A')$.

$F(A, A')$ can be evaluated by computing integrals over both of the patches for the function $g(x, x')$, i.e. computing the polygon-to-polygon form factor. But, to simplify the evaluation, it is possible to compute the portion of radiance from the centroid of the emitting patch arriving at the receiving patch, i.e. the point-to-polygon form factor. In addition, the direction-dependency of the outgoing radiance can be taken into account by using the directional

point-to-polygon form factor

$$F(x,\Gamma,A',\Theta)=\int_{A'}\mathcal{V}(x,\Gamma,x',\Theta)\frac{\cos(\gamma)\cos(\theta)}{|x'-x|^2}\,\mathrm{d}x', \quad (3)$$

where x is the centroid of the emitting patch, Γ is the solid angle through which the radiance is leaving that patch, A' is the receiving patch, Θ is the solid angle through which the radiance is received, and γ and θ are single incoming angles inside solid angles Γ and Θ, respectively. The visibility term $\mathcal{V}(x,\Gamma,x',\Theta)$ equals one when the line from x to x' is unobstructed and inside solid angle Γ on the emitting patch and inside solid angle Θ on the receiving patch, and zero otherwise.

3.1. Acoustic Radiance Transfer Method

Based on this formulation of acoustic energy transfer, a computational acoustic modeling method has been developed [1]. This acoustic radiance transfer technique models the transfer of acoustic energy in an enclosure. The geometric model of the room is divided into patches. Since it is possible to compute the energy transfer between two patches, the total energy transfer can be computed as follows.

First, the acoustic energy is shot from the sound source(s) to all unoccluded patches. The energy is weighted by the directional form factor and reflected at each patch according to the reflectance pattern of their material represented as a BRDF. The outgoing energy from each patch is stored on the patch for each direction (a set of directional slots is used).

Then, the patch with the highest undistributed energy is chosen and the energy stored on that patch is shot forward as if it were a sound source. The energy on that patch is marked distributed and the next patch with the highest undistributed energy is chosen. The process is repeated until the undistributed energy has fallen below a very small threshold value. The result of this phase is that the acoustic energy has been distributed on the surfaces of the room according to the geometry and the reflection properties of the materials. All the patches have stored the time-dependent outgoing energy in each direction.

The last phase of the technique is the final gathering, where the energy is collected from each unoccluded patch to the listener. This phase must be repeated every time the listener moves, while the previous phases can be precomputed.

3.2. Frequency Domain Operations

Since the acoustic radiance varies over time, the energy responses on the patches must be represented as functions of time. The main idea of the presented new method is to perform all the computations in the frequency domain. The time domain responses in the acoustic radiance transfer method can be replaced by frequency domain responses represented as discrete Fourier transform (DFT) coefficients a_k, where $k \in [0, N-1]$ for response length N. The coefficients are complex numbers, $a_k = x+y\mathrm{i} = r\mathrm{e}^{-\mathrm{i}\phi}$. The absolute value r of a coefficient corresponds to the magnitude of the frequency component while ϕ is the phase angle.

In the initial shooting the impulse originating from the source is a Dirac delta function. Conveniently, its DFT is very simple and all the coeffients of the initial response are equal to one.

To be able to perform the computation in the frequency domain, the frequency domain operations corresponding to the time domain operations utilized by the method must be found. The linearity of the Fourier transform assures that scalar multiplication and addition can be performed similarly to the time domain operations. The scalar multiplication is required for scaling the responses according to the form factors and BRDFs, and the addition is used in accumulation of the contributions of several responses into one response. In addition, time shifting is needed to implement the propagation delays. The relations between the required time domain and frequency domain operators are as follows:

$$\begin{aligned} & \mathcal{F} \\ \delta[n] &\longleftrightarrow 1 && (4)\\ ax[n]+by[n] &\longleftrightarrow aX(\mathrm{e}^{\mathrm{i}\omega})+bY(\mathrm{e}^{\mathrm{i}\omega}) && (5)\\ x[n-n_0] &\longleftrightarrow \mathrm{e}^{-\mathrm{i}\omega n_0}X(\mathrm{e}^{\mathrm{i}\omega}). && (6)\end{aligned}$$

3.3. Response Representations

For auralization purposes all the energy responses must be converted into equivalent impulse responses [20]. The samples in the energy response correspond to the intensity, which is proportional to the squre of the pressure, arriving at a certain area within a short interval determined by the length of the sample. The samples in the impulse response represent the pressure at a certain point and in a certain moment of time.

By making the same assumptions as when defining the form factor, a relationship can be derived between these two types of responses. The wavefronts arriving from one patch at another patch are then approximately planar, and the acoustic intensity can be written as

$$I = pu = \frac{p^2}{\rho c}, \quad (7)$$

where u is the particle velocity, ρ is the density of air, and c is the speed of sound. When examining the constant-length intervals of time, i.e. the sample length, it becomes apparent that the samples in the energy response are proportional to the square of the samples in the equivalent impulse response. Since DFT is linear, the same relation also holds in the frequency domain. A reasonable approximation of the equivalent impulse response to an energy response can be constructed by taking the square root of the magnitude of each sample and keeping the phases unchanged.

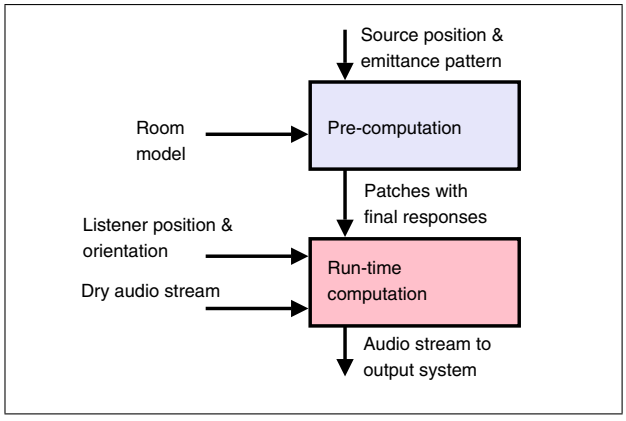

Figure 3. The simulation system for realtime auralization.

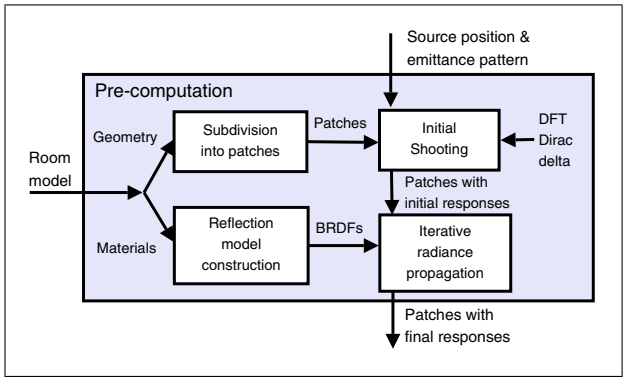

Figure 4. In the pre-computation phase, frequency domain directional responses to all the patches are computed.

4. Implementation

The presented novel auralization system consists of a pre-computation phase and a runtime computation phase, as illustrated in Figure 3. The acoustic radiance transfer is performed completely in the frequency domain, and the signals are transformed to the time domain only for final sound reproduction.

4.1. Pre-computation

In the pre-computation phase, shown in detail in Figure 4, the geometric model is subdivided into patches, the reflection models for all the materials are constructed, and the acoustic radiance transfer method is run to obtain the directional energy responses for all the patches.

4.1.1. Subdivision into patches

It is required that the 3-dimensional polygonal model of the room is split into patches. The simplest technique to make the subdivision into patches is to choose planes regularly along each coordinate axis and then cut the polygons by these planes. The patches do not have to be equal sized, but this is usually preferable so that the iterative propagation will converge faster. The run time of the precalculation phase is quadratically proportional to the number of patches, and thus very fine subdivisions may lead to excessively long execution times.

4.1.2. Reflection model construction

The effect of a boundary material on a sound wave is a complex one. Both the temporal and spectral behavior of reflected sound are functions of incident angle. However, in practice boundary materials are described only with plain absorption and diffusion coefficients at quite coarse frequency resolution, usually in octave bands. For these reasons, simplified angle-independent reflection models are usually applied in room acoustics modeling.

The obtained BRDFs are presented as frequency domain responses for incoming-outgoing direction pairs. The implementation with complex values enables even incorporation of phase behaviour into the reflection models, if such data would be available.

In the acoustic radiance transfer method the reflection directions can be represented as a hemisphere which is discretized into solid angles such that there is one frequency response in each solid angle. In the presented implementation, the discretization is quite coarse; the hemishpere is divided into 18 solid angles. Based on available absorption and diffusion data, the reflection models (BRDFs) are designed for energies, following the approach in [1].

4.1.3. Frequency domain acoustic radiance transfer

The acoustic radiance transfer, explained in Section 3.1, can be performed in the frequency domain by using the frequency domain operators for the responses. In the initial shooting an energy impulse, according to equation (4), is spread to all the visible patches. Then, as the energy is iteratively propagated between patches, the responses are scaled with form factors, utilizing equation (5), and time-shifted using the operator in equation (6). The frequency-dependent absorption by the materials and air can be directly applied to the samples in the frequency domain responses corresponding to the desired frequencies.

The results of the frequency domain computation, i.e. after the inverse DFT, are exactly the same as with the time domain computation [1] if the responses in the frequency domain computation use full resolution responses and not the compressed ones which are described below.

4.1.4. Compressed responses

With current hardware the length of the responses at patches has to be limited since the memory required for storing them is directly proportional to the response length. On the other hand, the quality of the response depends on its length. In the presented implementation, a compromise between the performance and quality was found with a response length of 1024.

Although, the pre-computed results are stored as vectors of 1024 samples, longer responses are needed for the computation. The inverse DFT of the 1024 samples long frequency domain response would produce the correct response in the time domain for a very low sampling rate or for a very short interval of time. For a sampling rate of 48 kHz a response of 65536 samples would be able to represent approximately a 1.4 s reverberation which is considered sufficient for this implementation.

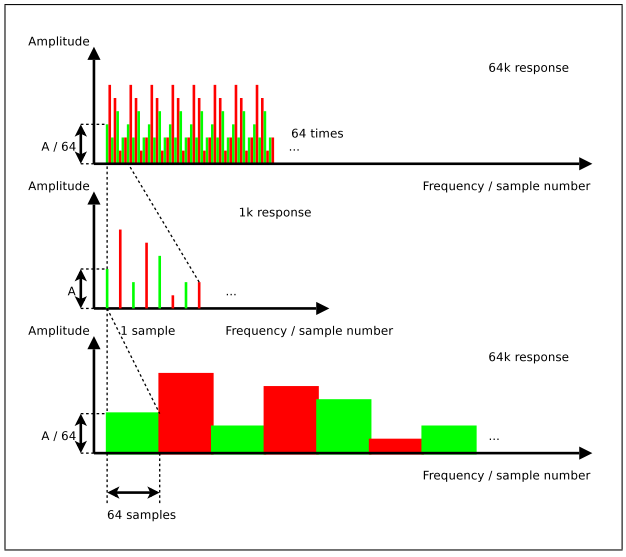

Figure 5. For the overlap-add convolution all the patch responses are upsampled from 1 k samples to 64 k samples. The original 1k response is in the middle. On top is a 64k response constructed by repeating the 1k response 64 times, and on bottom is a 64 response constructed by stretching the samples in the 1k response 64 samples wide.

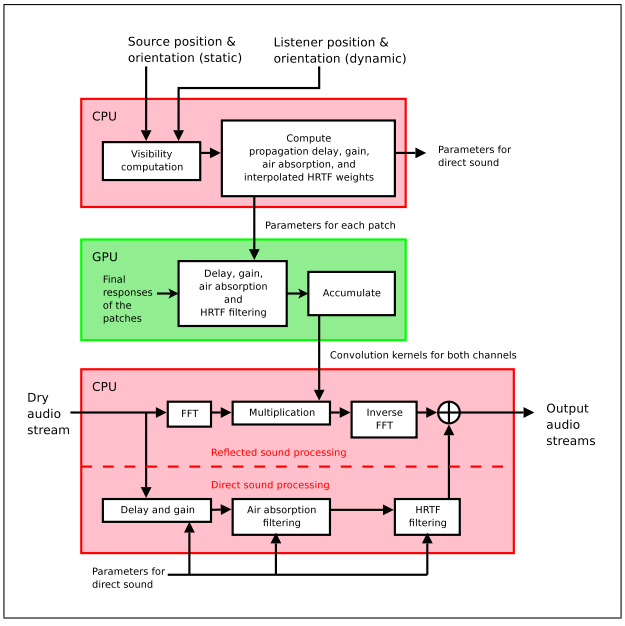

Figure 6. In the run-time processing, all the reflections are processed in the frequency domain while the direct sound is handled in the time domain.

There are several approaches to transform a 1024-sample response into a 65536-sample response. Simple interpolation would be possible by zero-padding the centre of the frequency domain response, but that would effectively remove the high frequencies, allowing only modelling of low frequencies, although the time structure would be correct. On the other hand, preserving the frequency content by upsampling the response in the frequency domain would destroy the time structure.

Thus the responses are upsampled using the upper one of techniques shown in Figure 5. Repeating the low-resolution response to fill the entire high-resolution response, preserves the time structure and produces the correct solution for low frequencies while the higher frequencies in the response are incorrect. However, the phase in the high frequencies will be incorrect in any case, since the dicretized patches are much larger than the wavelength and thus the phase can change dramatically over the surface of a patch, although only one phase value is used in the modeling process. According to informal listening tests, the auralized results still sound good, because the time structure is correct.

4.2. Run-time Computation

The run-time computation, outlined in Figure 6, consists of computation of parameters on the CPU, final gathering, including directional filtering, on the GPU, and finally convolution with a dry signal on the CPU. The final gathering, which constructs the filters for the convolution phase, is described in detail in the next section, but the CPU computation and the approach chosen for the HRTF filtering are covered in this section.

All the signal processing is performed in the frequency domain, except for the direct sound which is handled in the time domain on the CPU. In the presented implementation only headphone reproduction is considered, but the algorithm can be tailored for any multichannel reproduction system, e.g., vector base amplitude panning [21] or Ambisonics [22].

The parameter computation and the final gathering must be performed each time the listener moves. It is obvious that the achieved update rate depends on the complexity of the geometry.

4.2.1. Parameter computation

The first step is to find all the patches visible relative to the listener using a kd-tree structure, a hierarchical spatial data structure which can significantly accelerate the search process [23]. In the second step parameters such as the propagation delay and the attenuation factor (gain) are computed based on the distance from the patch. The air absorption is also included in the computation of the attenuation factor by using appropriate analytical formulations [24]. In addition, the incoming direction of the radiance from the patch to the listener is needed in directional filtering with HRTFs. Not the direction itself, but the interpolated HRTF weights (explained later) for each patch are given as parameters to the final gathering phase.

All these computations are also performed for the direct sound in a similar way than for the patches, but the parameters are passed directly to the direct sound computation on the CPU and not to the GPU.

4.2.2. Directional filtering for binaural output

Directional filtering is implemented by separating the interaural time delay (ITD) and the minimum-phase HRTFs

for 710 measured directions [25]. The ITDs can be added to the propagation delays already in the parameter computation phase. On the other hand, the direction dependent filters must be convolved with the responses of the patches. This can be done as a multiplication in the frequency domain.

Since a full measured HRTF filter database would take a large amount of space and they should preferably fit into the GPU memory, the database must be compressed. For that purpose, principal component analysis (PCA) is applied to the linear amplitudes of the minimum-phase HRTFs in the frequency domain [13]. Seven principal components account for 94% of the variance in the database, and using only them is considered a reasonable approximation. Thus, only seven responses are required to upload in the GPU memory in addition to the weighting factors for the measured directions.

Since the database contains ITDs and the coefficients for the PCA-components only for a limited number of directions, these parameters are interpolated bilinearly to get values for other directions. This results in smooth transitions. The interpolation is performed already on the CPU in the parameter computation phase and only the resulting weighting coefficients are passed as parameters to the GPU kernel.

In run-time, the interpolated coefficients are used for constructing the HRTF filters for the desired directions by simply summing up the weighted PCA components. Then, the processed response is multiplied with this filter in the frequency domain.

4.2.3. Convolution with the audio stream and final output

The method must be able to handle streaming input signal. Thus, the overlap-add method is used [26]. The stream is processed as overlapping segments, one segment at a time. Each segment is weighted by a windowing function. The overlapping windowing functions sum up to one, so that when the weighted segments are combined together after processing, the result is the same as if the whole original signal was processed at once.

The processing is performed such that the discrete Fourier transform (DFT) is taken of the weighted segment, thus producing the frequency domain segment. Then the result is multiplied by the filter produced in the final gathering phase. This modified segment is transformed back to time domain by the inverse DFT. The result is the original weighted signal convolved with the filter.

Care must be taken when combining the segments back together to produce the filtered output stream. The Blackman window, which requires overlap of $\frac{2}{3}$ of the segment length, is used. So at each processing step the stream progresses by $\frac{1}{3}$ of the segment length. Since the length of the result of the convolution is actually $L_s + L_f - 1$ samples, where L_s is the segment length and L_f is the filter length, the rest of the result (that is after taking off the first $\frac{1}{3}$ of the segment length) must be stored and summed together with the result of the next step. This final part of the result is called the overlap. The overlap must be carried through the process and updated at each step to correctly construct the final output.

The choice of the windowing function is crucial to the quality of the output signal. Since the frequency domain scaling of the low-sampling-rate responses into high-sampling-rate responses works properly only for periodic signals, and the input signal is aperiodic in general, there is an issue to solve. A sudden jump will occur, as after the last sample of the segment, the first sample of the next period is assumed to follow and that is exactly the first sample of the current segment. When approximating a sudden jump with a finite number of frequencies, as in the DFT, the so called Gibb's phenomenon appears, which means that there will be unwanted ripples near the jump no matter how many frequencies are used, as long as the number is finite. Thus, the first and last part of the segment are deteriorated by these ripples. A solution is to window the signal so that the apparent periodic signal does not contain any sudden jumps. That can be done by forcing the derivative of the signal to be continuous at the first and last sample of the segment. Since the derivative of the signal

$$g'(s) = f'(s)w(s) + f(s)w'(s), \tag{8}$$

and the input function $f(s)$ can be arbitrary, both the windowing function $w(s)$ and its derivative $w'(s)$ must be zero at the end points of the segment. The Blackman-window,

$$w(s) = \frac{1-\alpha}{2} - \frac{1}{2}\cos\left(\frac{2\pi n}{N-1}\right) + \frac{\alpha}{2}\cos\left(\frac{4\pi n}{N-1}\right), \tag{9}$$

satisfies these conditions.

There are a few implementation details which set limits to what can be achieved. First, the FFT block size greatly affects the running time. In the presented implementation, a size of $2^{16} = 65536$ samples was chosen since with this length it was still possible to perform the FFTs in real time while the length of the response is still sufficient for 1.4 s of sound at 48 kHz. This is sufficient assuming that the reverberation is insignificant after 1.4 s. For small rooms with faster sound decay, an even shorter FFT block could be used but for more reverberant spaces this length is the minimum requirement. Secondly, the window size, L_s, affects the delay introduced before output. A small size is desirable, thus 8192 samples were used. Since the windows are overlapped, the maximum delay between a change in the parameters and the corresponding effect in the output is $\frac{1}{3}L_s$. The playback rate is 48 kHz, thus a delay of $(8192/3)/48\text{kHz} \approx 56.9\text{ms}$ is caused. A smaller window size could be used, but even now the 64 kB FFT must be performed $48000/(8192/3) = 17.58$ times per second, and the number of FFT executions is inversely proportional to the window size.

4.2.4. Direct Sound

As mentioned earlier, the direct sound needs special treatment. The overlap-add convolution does not fit perfectly for a dynamic scenario where, e.g., the listener moves. In

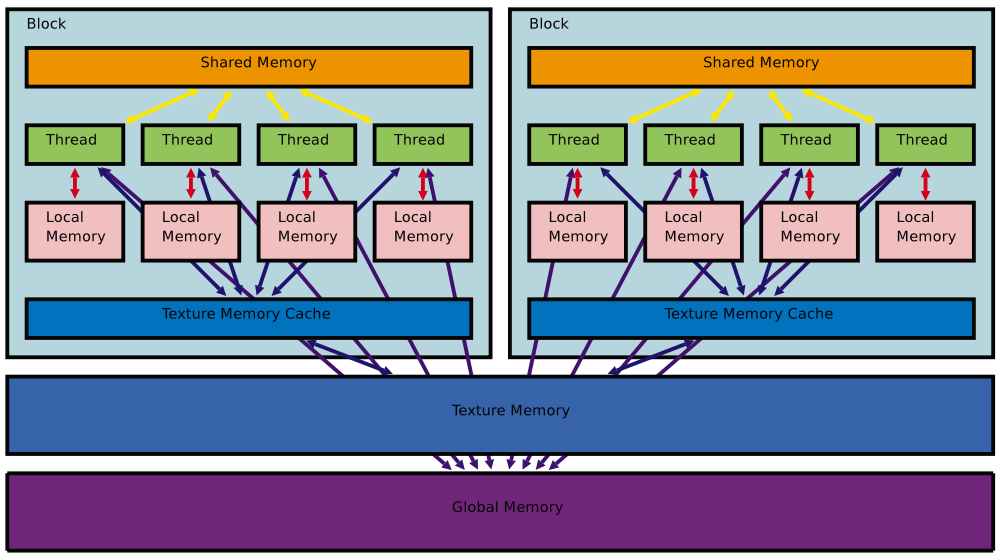

Figure 7. Local, shared, texture, and global memories on the GPU.

practice, moving the listener with the overlap-add technique results in clearly audible distortion in the output signal. This is due to the fact that propagation delay changes only at the rate corresponding to third of the size of the window applied in convolution, whereas it should be changed on a per-sample basis to gain distortion-free output. The problem could be reduced by using an even smaller window and bigger overlap. This turns out to be computationally too expensive, and a separate time-domain processing for the direct sound is implemented.

Movements are implemented with a fractionally addressable delay-line, and the HRTF's are implemented as minimum-phase FIR filters with separate delay-lines for ITD. The approach is exactly the same as in the DIVA system [5]. Based on informal listening experiences, it is sufficient to process only the direct sound separately, and the distortion in dynamic reflections is not audible.

4.3. Parallel Computation with Graphics Hardware

The run-time final gathering implementation is built on the CUDA-library by NVIDIA [27]. Other approaches would be possible, but since a suitable application programming interface is available for hiding the device-specific details, that is utilized. The computational model is based on threads which work in parallel, although in reality some of the computation might be sequential if there are not sufficiently many multiprocessors available. A multiprocessor consists of a set of processors executing the same set of instructions but with different data. One multiprocessor can run even hundreds of threads concurrently and there can be several multiprocessors on a graphics card. The details depend on the specific device used, but the number of threads is large enough to call the processing massively parallel.

The threads are grouped into blocks in which they can be synchronized and which can share some variables. Synchronization between the blocks is not possible and data sharing can happen only through global memory. One block consists typically of a few hundred threads. In the presented implementation the block size is 256 threads.

4.3.1. GPU vs. CPU processing

In the presented implementation most of the final gathering [1] process is performed on the GPU. The initial shooting and iterative propagation phases are precomputed on the CPU. The input of the GPU algorithm consists of the responses on the patches and the parameters which tell which of them are visible, what is the delay of sound traveling from the patch to the listener, what is the attenuation by the distance and what are the direction-dependent weighting factors for the head-related transfer functions. The output consist only of the filters for both left and right channel. These filters are then convolved with a dry audio stream on the CPU to produce the final output singals.

It should be noted that the whole process in performed in the frequency domain to allow fast convolutions (implemented as multiplications on the GPU). Only the final signals are constructed on the CPU by taking a fast inverse Fourier transform of the output signals.

4.3.2. Memory arrangements

To be able to write efficient code for the GPU, it must be taken into account that there are different types of memories on the GPU. Each processor of the multiprocessor may have a set of registers. These can be used for the *local* variables in the GPU kernel code. The reads and writes are very fast. Then some of the memory can be *shared* among the processors working on the same block of threads. However, syncronization is required at this level to avoid overlapping reads and writes, which results in slower access times in some cases.

Some of the memory is *read-only*, i.e. it cannot be written by the processors. This memory can be cached and it is often used for storing textures in graphics applications. Thus, the spatial locality in two dimensions affects

the access time, since the cache entries are read in two dimensional blocks of "texture". *Global* memory is memory which the processors can both read and write, but at very slow speed, typically hundreds of times slower than operating on the local memory. Figure 7 shows the memory arrangements on the GPU.

Optimizing memory accesses

The global memory is used for the 64k output filters for each channel, since that is the only memory all the threads can write. To avoid accesses to this slow memory, the filters are constructed in blocks of 256 samples which are written to the output at once and only once. Before writing out the block, the temporary storage for each 256-sample block is in the shared memory.

The shared memory is accessed by multiple threads so that each thread operates on a different sample to avoid conflicts. Each thread is assigned to a patch. First, each thread adds the effects from their patch to one sample. When a thread finishes operating on a sample it reaches a synchronization point where it has to wait for the other threads to finish their operations. Then at the same time, all threads move one sample forward and add the effects of their patch to the next sample. This process is repeated until all the threads have added the effects of their patch to all the 256 samples. When a block of threads has processed all its 256 samples, it writes them out to the output filters in the global memory. Since the filters are 64k long, 64k / 256 = 256 blocks of threads are required.

On the other hand, in the presented implementation a block may contain a maximum of 256 threads, so if there are more than 256 patches, the whole process (GPU kernel code) must be run more than once, changing the patches assigned to the threads each time. If all the responses of the patches can fit into the GPU texture memory, they can be loaded there in the beginning. Otherwise, the GPU texture memory works as a cache to the computer's random access memory which contains all the responses, and its contents are changed between calls to the GPU kernel code.

4.3.3. Delay, attenuation, air absorption, HRTFs

The delay-operator applied at run time is based on the distance from the source patch to the receiver. In addition, the interaural time delay must be taken into account. These are given as input parameters and applied at once. The delay-operator in the frequency domain simply becomes multiplication by $e^{-i2\pi fn}$, where f is the frequency and n is the integer delay in samples.

Attenuation follows the inverse squared distance law, and it could be applied directly by multiplying the response with the attenuation factor, but in the presented implementation it is factored in the weighting coefficients for the head-related transfer functions in the parameter computation phase.

Air absorption is dependent on the distance and frequency. Since the dependency is complex, it cannot be easily computed in real time. Thus, filters for air absorption are designed beforehand for distances from 1 meter

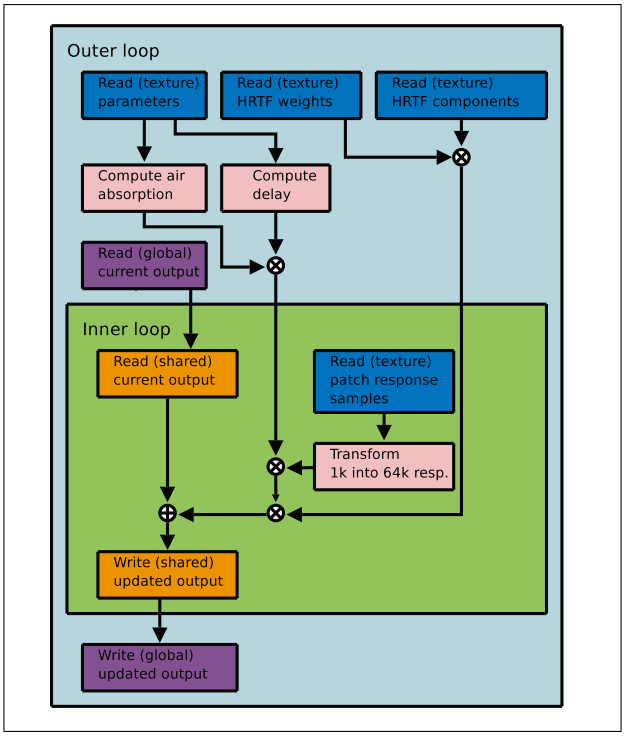

Figure 8. The computation pipeline in the GPU kernel.

to 80 meters with one meter step. This accuracy is sufficient as the transitions are smoothed out by the windowing function used in the signal processing part of the computation. The maximum distance is also sufficient in typical cases, such as concert halls, where the longest free paths are much less than 80 metres. The filters are stored on the GPU's memory and they are applied in realtime by using the distance as an index to the look-up table.

Since these operations are the same for every sample, they must be done only once per block of samples. Figure 8 illustrates the process. The outer loop is run once per block of samples, but the inner loop must be run for every individual sample. The parameters can be read in the outer loop since they are the same for the whole response. Then it is possible to compute the effects of air absorption and delay (including ITD) and combine them. Also the HRTF weights and components can be read in the outer loop and the HRTFs can be constructed. The attenuation by distance is already factored in the HRTF component weights on the CPU. Reading and writing to the global memory is done in blocks in the outer loop while the intermediate results are kept in the shared memory. The output buffers in the global memory must be read and written to the shared memory since they can contain temporary results if the number of patches is so high that the GPU kernel must be run more than once. In the inner loop the patch response samples are read from the texture memory and transformed into equivalent frequency domain impulse responses as described in Section 4.1.4. The result is multiplied with the intermediate results constructed in the outer loop and finally the results are accumulated to the intermediate results. After processing all the threads the temporary results in the

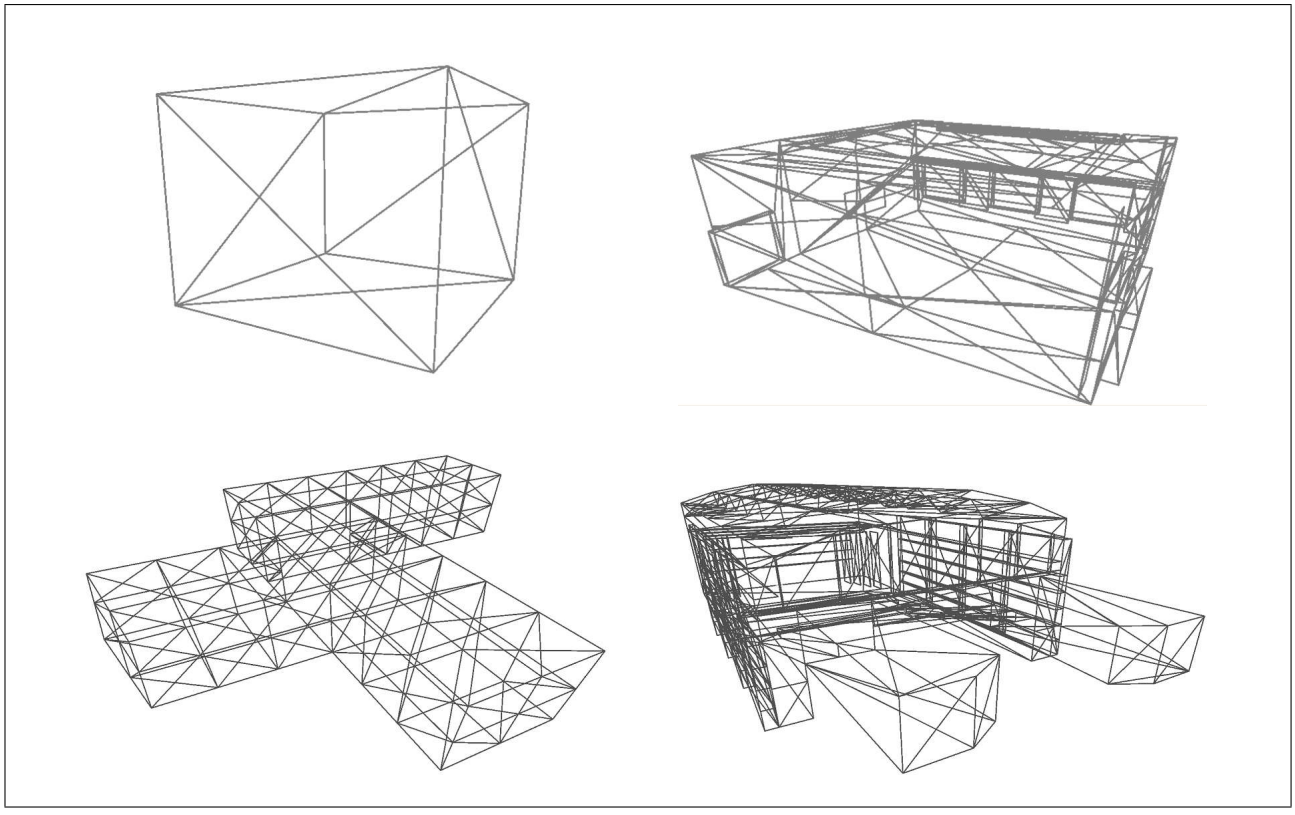

Figure 9. The performance of the system was evaluated in four different room geometries: a cube (12 patches), a corridor model (256 patches), a small hall model (243 patches) and a concert hall model (1176 patches).

shared memory are written to the global memory in the outer loop.

5. Results and Discussion

The implementation of the frequency domain acoustics radiance transfer method which utilizes the GPU was run to assess the performance and the quality of the auralization. The goal was to be able to update the output interactively while the listener was moving, while the perceived sound quality would be comparable to that achieved by an offline algorithm which has been shown to produce good results [1].

5.1. Performance

The performance of the system was tested with four test models. The cube model consists of 12 patches, the corridor model of 256 patches, the small hall model of 243 patches, and the concert hall model of 1176 patches. The models are illustrated in Figure 9. Visually relatively plain models were used since in practice, small details do not cause audible effects. In the walkthrough a more detailed model can be utilized for the visualization while a reduced model is used for the acoustics modeling.

The tests were run on a desktop PC with a 2.0 GHz Intel Core 2 Quad processor, 2 GB of memory, and Nvidia GeForce 8800 GTX display driver.

Table I. Pre-computation times for the test models. Ini.S.: Initial Shooting [s].

Model	Patches	Ini.S.	Iter. Propagation
Cube	12	0.03	14 min 56 s
Corridor	256	0.99	10 h 7 min 17 s
Small Hall	243	1.02	3 h 7 min 5 s
Concert Hall	1176	7.15	99 h 12 min 32 s

Table I shows the pre-computation times for the test models. The number of iterations was fixed to be one hundred times the number of patches. It can be seen that the pre-processing is quite a slow operation, in particular when the number of patches grows.

The run-time computation times are listed in Table II. The time used for accumulating the responses, the total time used by the GPU, i.e. the time used for both accumulation and memory transfers, and the parameter update rate are shown. Since the blocks in the overlap-add convolution and the parameters for the direct sound are updated at the rate of 17.6 Hz, that is the minimum parameter update rate required for updates for every block. Thus, the performance is sufficient in the case of the three simplest models. But for the most complex model, the filters are updated approximately only once per three blocks. The latency is around 160 ms which is even then of the same magnitude than in other interactive walkthrough systems [5, 28].

Table II. Run-time computation times for the test models. Accu.: Accumulation [ms], GPU: GPU total [ms], Update: Update rate [Hz].

Model	Accu.	GPU	Update
Cube	4.31	26.3	30.1
Corridor	11.2	40.4	19.8
Small Hall	12.7	43.0	19.5
Concert Hall	61.5	138	6.10

Although, the obtained overall performance is sufficient for the shown test cases, they are still rather simple when compared to a typical geometry applied in global illumination. If the same models should be used for both visual and aural rendering, an additional geometry simplification should be added to the pre-processing to make realtime auralization possible. The level of details used in the test case models in this study is sufficient, since small details are not acoustically significant.

5.2. Quality

Since the emphasis of this work was on achieving high performance, arranging formal listening tests to evaluate the quality was not considered necessary. Informal listening confirmed that the perceived sound quality was good in general, but with the large concert hall model some artifacts were present due to the slow update rate.

The accuracy of the final result depends mainly on two factors, first being the accuracy of the acoustic BRDFs. Currently available acoustic material data is quite coarse and limits the accuracy of reflection modeling. The second factor is the applied discretization such that the use of smaller patches, smaller solid angles in angular discretization, and longer patch responses would improve the quality of the result. However, since these factors and the computational framework are the same as in the time domain acoustic radiance transfer method, the results correspond to those achieved by [1], whose work was validated against measured data [29, 30]. The small hall model used in the performance tests is the model of the room used in that validation.

The response length in the time domain method was 1024 samples which was sufficient for extracting acoustic parameters [1]. The responses for the patches where generated for each octave band separately. The frequency domain implementation with the same patch response length, would produce the same results where the patch responses of different octave bands are combined in one response. However, the requirements for the output are different for the auralization than for parameter extraction. Temporal energy distribution is not sufficient, but an equivalent frequency domain impulse response must be constructed. As mentioned in Sections 3.3 and 4.1.4 this process causes some error, but since the acoustical parameters computed from the patch responses would be realistic, the auralized output signal gives a realistic impression of the space.

6. Limitations and Future Work

The presented system is efficient enough for interactive walkthroughs, although in many ways it is not perfect. First of all, it is obvious that in reality diffraction occurs, although it has not yet been modelled in the current system. At low frequencies, the effects of the diffraction should be audible. However, the framework of the acoustic radiance transfer does not hinder the implementation of diffraction, indeed, it seems that Biot-Tolstoy-Medvin solution [31] could be applied [32].

Another drawback of the presented technique is the stationary sound source. When the source moves, the precomputation must be redone, and this is a time-consuming process. Multiple sound source locations could be applied in the system, but the acoustic radiance transfer preprocessing must be carried out separately for each of them. Finding ways to overcome this limitation needs further research.

As more and more accurate models are used the memory consumption will increase, and it might appear necessary to study more advanced compression techniques such as use of spherical harmonics [33] in the presentation of directional responses.

7. Conclusions

The acoustic radiance transfer method was reviewed and it was noted that the acoustic energy responses could be represented in the frequency domain and that operations corresponding to emitting an impulse, as well as scaling, adding, and delaying the responses in the frequency domain was possible. This made possible a frequency domain implementation of the technique.

The results of the most time-consuming part of the computation, the iterative propagation of the acoustic energy, can be stored, and thus only the final part of the computation must be re-run when the listener moves. This final part, involving operations on a large number of responses, can be implemented on the graphics hardware which allows utilization of massive parallellism where one thread handles one sample of a response. The implementation tries to minimize the slow memory access which otherwise could form a bottleneck in the process.

The limits of the memory space and the structure of the GPU pipeline forced to certain implementation solutions. The head-related transfer functions were compressed using principal component analysis so that they could fit in the memory of the graphics card. Bilinear interpolation of the weights of the components between the measured directions was used to produce smooth directional transitions. The overlap-add technique with the Blackman window was used for the audio stream processing. The direct sound was handled separately since the overlap-add technique does not allow per sample, but per block, updates for the parameters.

The results show that interactive walkthroughs in relatively complex models with arbitrary reflection properties

is possible while using a physically-based model for the whole room response.

Acknowledgements

This work was partly supported by the Finnish Funding Agency of Technology and Innovation (project 3Dr) and the Academy of Finland (grant 119092).

References

[1] S. Siltanen, T. Lokki, S. Kiminki, L. Savioja: The room acoustic rendering equation. Journal of the Acoustical Society of America **122** (Sep 2007) 1624–1635.

[2] T. Funkhouser, J.-M. Jot, N. Tsingos: "Sounds good to me!" Computational sound for graphics, virtual reality, and interactive systems. SIGGRAPH'02 Course Notes, 2002. Also available as http://www.cs.princeton.edu/~funk/course02.

[3] T. Takala, J. Hahn: Sound rendering. SIGGRAPH Comput. Graph. **26** (1992) 211–220.

[4] M. Kleiner, B.-I. Dalenbäck, U. P. Svensson: Auralization – an overview. Journal of the Audio Engineering Society **41** (Nov 1993) 861–875.

[5] L. Savioja, J. Huopaniemi, T. Lokki, R. Väänänen: Creating interactive virtual acoustic environments. Journal of the Audio Engineering Society **47** (Sep 1999) 675–705.

[6] D. Schröder, T. Lentz: Real-time processing of image sources using binary space partitioning. Journal of the Audio Engineering Society **54** (July 2006) 604–619.

[7] J. Blauert: Spatial hearing. the psychophysics of human sound localization. 2nd ed. MIT Press, Cambridge, MA, 1997, 494.

[8] J. Mackenzie, J. Huopaniemi, V. Välimäki, I. Kale: Low-order modeling of head-related transfer functions using balanced model truncation. IEEE Signal Processing Letters **4** (1997) 39–41.

[9] D. W. Batteau: The role of the pinna in human localization. Proceedings of the Royal Society of London Series, 1968, 158–160.

[10] K. Genuit: A description of the human ear transfer function by elements of communication theory. Proceedings of the 12th International Congress on Acoustics, Toronto, Canada, 1986.

[11] J. Chen, B. D. V. Veen, K. E. Hecox: External ear transfer function modeling – a beamforming approach. Journal of the Acoutical Society of America **92** (1992) 1933–1944.

[12] W. Martens: Principal components analysis and resynthesis of spectral cues to perceived direction. Proc. Int. Computer Music Conf. (ICMC'87), University of Illinois at Champaign/Urbana, USA, 1987, 274–281. Also available as http://www.music.mcgill.ca/~wlm/papers/Martens_1987_ICMC.pdf.

[13] D. J. Kistler, F. L. Wightman: A model of head-related transfer functions based on principal components analysis and minumun-phase reconstruction. Journal of the Acoustical Society of America **91** (March 1992) 1637–1647.

[14] J. C. Middlebrooks, D. M. Green: Observations on a principal component analysis of head-related transfer functions. Journal of the Acoutical Society of America **92** (1992) 597–599.

[15] J. Chen, B. D. V. Veen, K. E. Hecox: A spatial feature extraction and regularization model for the head-related transfer function. Journal of the Acoutical Society of America **97** (1995) 439–450.

[16] M. J. Evans, J. A. S. Angus, A. I. Tew: Analyzing head-related transfer function measurements using surface spherical harmonics. Journal of the Acoustical Society of America **104** (1998) 2400–2411.

[17] E. Gallo, N. Tsingos: Efficient 3d audio processing on the gpu. Proceedings of the ACM Workshop on General Purpose Computing on Graphics Processors, August 2004, ACM. Also available as http://www-sop.inria.fr/reves/Basilic/2004/GT04.

[18] N. Tsingos, C. Dachsbacher, S. Lefebvre, M. Dellepiane: Instant sound scattering. Proceedings of Eurographics Symposium on Rendering, 2007, Eurographics. Also available as http://vcg.isti.cnr.it/Publications/2007/TDLD07.

[19] N. Röber, M. Spindler, M. Masuch: Ray acoustics using computer graphics technology. Proceedings of the 10th International Conference on Digital Audio Effects (DAFx), Bordeaux, France, September 2007, 117–123. Also available as http://dafx.labri.fr/papers/p117.pdf.

[20] K. H. Kuttruff: Auralization of impulse responses modeled on the basis of ray-tracing results. Journal of the Audio Engineering Society **41** (November 1993) 876–880.

[21] V. Pulkki: Virtual sound source positioning using vector base amplitude panning. Journal of the Audio Engineering Society **45** (June 1997) 456–466.

[22] D. G. Malham, A. Myatt: 3d sound spatialization using ambisonics techniques. Computer Music Journal **19** (1995) 58–70.

[23] H. Samet: The design and analysis of spatial data structures. Addison-Wesley, 1990.

[24] H. Bass, H.-J. Bauer: Atmospheric absorption of sound: Analytical expressions. Journal of the Acoustical Society of America **52** (1972) 821–825.

[25] W. G. Gardner, K. D. Martin: HRTF measurements of a KEMAR. Journal of the Acoustical Society of America **97** (Jun 1995) 3907–3908.

[26] J. B. Allen, L. R. Rabiner: A unified approach to short-time fourier analysis and synthesis. Proc. IEEE **65** (Nov 1977) 1558–1564. Also available as http://ieeexplore.ieee.org/iel5/5/31259/01455039.pdf.

[27] NVIDIA: Compute unified device architecture (CUDA) - library. ONLINE, 2008. http://www.nvidia.com/object/cuda_home.html.

[28] T. Lentz, D. Schröder, M. Vorländer, I. Assenmacher: Virtual reality system with integrated sound field simulation and reproduction. EURASIP Journal on Advances in Signal Processing **2007** (2007). Article ID 70540, 19 pages.

[29] I. Bork: Report on the 3rd round robin on room acoustical computer simulation - Part I: Measurements. Acta Acustica united with Acustica **91** (2005) 740–752.

[30] I. Bork: Report on the 3rd round robin on room acoustical computer simulation - Part II: Calculations. Acta Acustica united with Acustica **91** (2005) 753–763.

[31] U. P. Svensson, R. I. Fred, J. Vanderkooy: Analytic secondary source model of edge diffraction impulse responses. Journal of the Acoustical Society of America **106** (1999) 2331–2344.

[32] S. Siltanen, T. Lokki: Diffraction modeling in acoustic radiance transfer method. Proceedings of Acoustics'08, Paris, France, June 29-July 4 2008, 6495–6500.

[33] J. Kautz, P.-P. Sloan, J. Snyder: Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering, Aire-la-Ville, Switzerland, Switzerland, 2002, Eurographics Association, 291–296.