Real-time impulse response estimation based on mirrored virtual sound sources

Sadjad Siddiq
Audio Researcher, Advanced Technology Division, Square Enix Co. LTD., 6-27-30 Shinjuku, Tokyo, Japan
E-mail: siddsadj@square-enix.com

Abstract. A fast method to estimate the impulse response of a fixed sound source to a listener at a movable location in a room of arbitrary shape in two dimensions is proposed. The method can be used in video games to calculate the reverberation of sounds placed in a level depending on the position of the listener and can also be used in 3d environments in which the listener mainly moves in two dimensions. After simplifying the geometry of the room using clustering and converting it to a 2d map, early reflections are modelled in detail using an enhanced image source method that takes occlusion into account: the position of the sound source is mirrored on all walls that are hit directly by its sound and virtual sound sources are created at the mirrored locations. This step is repeated recursively to calculate higher order reflections. Based on this pre-computation and the position of the listener, an impulse response of the early reflections in the room is generated in real time. The reflection data is enhanced stochastically using noise bursts to yield a more convincing acoustic effect. The late reflections in the tail of the impulse response are estimated based on the earlier part of the impulse response.

1. Introduction
Typical levels of role playing games in which players can move through three dimensional space usually feature a lot of different acoustic environments like small rooms, big halls or open space. Rendering the sound in such environments in a realistic way increases the immersiveness of the player in the game. Additionally sound that is spatialized in a realistic way can also serve as an additional cue for the player, carrying information about the location of the sound source. Especially in virtual reality applications the realistic spatialization of sound is a very important illusion to trick the player’s brain into believing it is actually inside the virtual environment.

Methods used to calculate the propagation of sound can be classified into two categories: methods based on solutions of the wave equation and methods based on geometrical acoustics. A notable example of a method based on solutions of the wave equation by Raghuvanshi and Snyder [2] allows to render sound with high quality at a high memory cost (around 100 Mb of compressed data needed for large game maps) and long computation time, which is typical for such solutions.

Savioja et al. [1] give a detailed overview over methods based on geometrical acoustics. Geometrical methods are in general much faster than methods that are based on solutions of the wave equation but they have the disadvantage that diffraction and scattering of waves, which play an important role in sound propagation, are neglected since they are not described by the underlying models that are based on specular reflections. To take diffraction and scattering
into account, they have to be implemented on top of the system based on geometrical acoustics. Methods of geometrical acoustics include ray tracing ([3], [4]), beam tracing ([5]), image sources ([8], [10], [11], [12]), surface based techniques like radiosity ([6], [7]) or hybrids, for example of image sources and beam tracing ([9], [13], [14]). The method presented in this paper is also a combination of the image source method and beam tracing.

In the image source method a sound source is mirrored at reflecting surfaces by placing a virtual sound source at a mirrored location at the other side of the surface. This process is repeated for all reflecting surfaces. After mirroring the sound source at all surfaces, the virtual sound sources created in this process are again mirrored at reflecting surfaces. This process can be repeated until an appropriate number of virtual sound sources are found. The virtual sound sources are usually calculated offline. During runtime the time of the incoming reflections at any point in space can be calculated based on the distance between this point and the virtual sound sources [8].

The basic algorithm does not include any mechanism to check if sound from a virtual sound source actually reaches the point of the listener or if the reflection path is obstructed. In an early implementation based on the image source method and beam tracing that can be applied to arbitrarily shaped rooms by Borish reflections paths are evaluated at runtime to check if they reach the listener [9].

The method proposed by Funkhouser et al. also checks if reflections paths lead to the listener without obstruction and they can be used for complex environments, because looking up beams is implemented in an efficient way by subdividing the environment into cells. [14]

There are also some solutions that take diffraction into account. Mechel added additional virtual sources in corners to account for diffraction and reduced the environment to two dimensions for some evaluations for efficient implementation. [10] Stephenson proposed an implementation in which new beams are inserted into the simulation at certain points to take the diffraction of sound waves around corners into account. Beams are grouped together for efficient computation. [13]

In this paper we describe a method that uses image sources and beam tracing to calculate the early reflections of the impulse response of a fixed sound source position to a movable listener based on simplified geometry of the environment. Diffraction and diffusion are not taken into account during these simulations but the obtained impulse response is enriched stochastically. Informal listening tests have confirmed that the enriched impulse response leads to more realistic results than the plain impulse response.

2. Overview
The goal of the method presented in this paper is to find the impulse response $h$ that captures the acoustic effects impacted on sound travelling from an immobile sound source at point $S$ to the position of the player $L$. Direct sound and early reflections are calculated based on the image source method, while late reflections are estimated based on the calculated early reflections. The sound source is reflected at the walls of the level and virtual sound sources are created for all reflections. Occlusion is taken into account by limiting the area in which each virtual sound source is valid. These calculations are preformed before the execution of the game.

During game execution the player's position is compared to the areas in which the sound sources are valid. This is done for each sound source. If the player is in an area where a virtual sound source is audible, the reflection corresponding to that sound source is added to the impulse response $h$. After calculating all early reflections, the late reflections are estimated based on the amplitude and decay of the early reflections. The impulse response $h$ is then applied to the sound played at location $S$ to give the acoustic impression that the sound was actually played at position $S$ in the level.
3. Simplification of Geometry

In modern games meshes describing the geometry of a level tend to be very complex. This complexity is a disadvantage for our method, because the sound source is mirrored at every wall in the level. To make this process faster, we simplify the geometry. The level is first converted to a 2d map and then the 2d map is simplified. This is done by taking a slice through the geometry at the height of the head of the player. This is done by first calculating a surface parallel to the ground the player walks on at the height of the head of the player. We then only consider those triangles or rectangles of the mesh that touch or cross this surface. These triangles or rectangles are simplified to two-dimensional lines by discarding the 3rd dimension of their vertices. Lines with zero length are also discarded.

In the following step the geometry of the 2d map is simplified. If the distance between a point and its closest neighbour is under a certain threshold the mean of the point and the neighbour is calculated and this new point then replaces the point and its neighbour. This algorithm is iterated repeatedly over all points until no pairs of points exist whose distance is under the threshold. See figure 1 for an example. Note that the lines in the simplified geometry are not necessarily parallel to the lines in the original geometry, but this discrepancy is negligible because the exact direction of the reflections calculated in the next step is not important.

![Figure 1. Slice through original complex geometry (left side; 5646 points, 18760 lines) and simplified geometry (right side; 64 points, 123 lines)](image1)

**Figure 1.** Slice through original complex geometry (left side; 5646 points, 18760 lines) and simplified geometry (right side; 64 points, 123 lines)

![Figure 2. The sound propagating as spherical waves from sound source S is reflected at an obstacle (thick line). The reflected sound can be thought of as coming from point S', which is a mirrored version of S](image2)

**Figure 2.** The sound propagating as spherical waves from sound source S is reflected at an obstacle (thick line). The reflected sound can be thought of as coming from point S', which is a mirrored version of S

4. Virtual sound sources

The sound emitted by a sound source at point $S$ propagates through space as a spherical wave until the wave hits an obstacle and is reflected by this obstacle. The part of the wave being reflected at the obstacle can be represented as a wave coming from point $S'$, which is point $S$ mirrored at the surface of the obstacle as can be seen in figure 2. The reflected waves now continue to travel through space until they in turn hit an obstacle and are reflected once again. This reflection can also be represented by calculating spherical waves emitted from a virtual sound source $S''$ that is a mirrored version of $S'$, using the surface of the reflecting obstacle as mirror axis. This process continues until the energy of the sound wave is completely absorbed by the reflecting surfaces or by the medium the wave is travelling in. Note that at this stage we do not consider wave diffraction and scattering.

In two dimensional space, the wave propagates as a circular wave around $S$ and obstacles are modelled by line segments. The propagation of circular waves around point $S$ is calculated in
beams represented by circular sectors. Cutting the circle around $S$ in circular sectors facilitates the task of detecting which obstacles are hit by the circular wave, because the wave front is cut into small portions that can be examined independently by looking only into the direction of the beam. Each beam is thus represented by a circular sector defined by the center point $Q$ of its underlying circle, the angles $\omega_1$ and $\omega_2$ and their minimum and maximum radius $r_1$ and $r_2$. The radii represent the distance from the virtual sound source $Q$ between which the beam is valid.

Assuming that a beam is reflected by the line segment between the points $P_1$ and $P_2$, a new circular sector from point $Q'$, which is point $Q$ mirrored at the line $P_1P_2$, is created to represent the reflected beam. The beam is only valid from the point where it hits the obstacle and is reflected. This information is stored in $r'_1$, which is set to the distance between the new center point $Q'$ and line segment $P_1P_2$, calculated as $r'_1 = \min(QP_1, QP_2)$, where $\min$ returns the smallest value of its arguments. The beam that was reflected is terminated at this distance, so $r_2$ is set to the same value. We then check if the new beam hits any obstacle to find its value for $r'_2$.

Figure 3. A circular sector hits an obstacle represented by a line segment and is reflected partially.

Figure 4. Flow chart of the algorithm used to find the intersection(s) of two beams. There can be 0 to 2 intersections. The intersections are represented by $\alpha_1$, $\alpha_2$ and $\beta_1$, $\beta_2$. If values are not set no intersection is found. {} means that a possible solution previously considered is discarded. The function $\min$ and $\max$ return the smallest or greatest value of their arguments.

To check if a beam starting at radius $r_1$ hits an obstacle represented by the line segment $P_1$ and $P_2$, first the distances between $Q$ and $P_1$ and $P_2$ are calculated. If the distance to one of the points is greater than $r_1$ and the line segment lies in the direction of propagation of the beam, it is hit. To find out if the beam propagates towards the line segment, we first calculate the angles between the x-axis and the vectors $QP_1$ and $QP_2$ as $\phi_1$ and $\phi_1$ respectively, which can also be interpreted as a beam from $Q$ to the line segment. We then compare the angles of the actual beam representing the wave and the beam to the line segment according to the flow chart in figure 4 to find out if the wave beam hits the line segment. There are three cases: (1) the line segment is not hit, (2) the beam hits the line segment and is completely reflected, (3) the beam hits the line segment and is partially reflected. In case (3) only the portion of the beam hitting the obstacle is reflected. Such a case as shown in figure 3. Only the beam between angles $\phi_1$ and $\phi_2$ is reflected. The beam is then split into two sectors. One sector in whose direction the
wave continues to travel and one sector which terminates at radius \( r \), which is the distance to line segment \( P_1P_2 \). The beam is then reflected at the obstacle as described above.

Note that a beam being partially reflected can also be split into three parts, if the line segment reflects only the center portion of the beam, leaving both edges intact. This is why the algorithm in the flow chart in figure 4 returns up to two angle pairs: \( \alpha_1, \alpha_2 \) and \( \beta_1, \beta_2 \).

After examining all beams, the newly created beams are examined and it is checked if they hit an obstacle. If so they are again reflected after splitting them into a reflected and a non-reflected part if necessary. This algorithm is repeated until the sound has travelled for a period longer than the time for which we want to calculate early reflections.

Every beam is thus represented by a circular sector and stands for a reflection of the sound. In order to synthesize an impulse response we also store the number of reflections \( N_R \) the wave has undergone until it reaches the area described by the circular sector for each beam.

5. Calculation of early reflections

Based on the pre-calculated circular sectors, the early reflections at any point \( L \) in the level can be calculated. For every circular sector, we check if the point \( L \) lies between the angles \( \phi_1 \) and \( \phi_2 \) and if its distance from the center point of the circular vector is between \( r_1 \) and \( r_2 \). If all of these conditions are true, the reflection represented by the circular vector is added to the impulse response containing the early reflections.

A simple way to model reflections is by adding an impulse at time \( r/c \), where \( c \) is the speed of sound in air and \( r \) the distance between the center point of the circular sector \( P \) and \( L \). To generate a multichannel output signal, panning can be applied to the impulse based on the angle between the orientation of the player and the direction of the virtual sound source. The amplitude of the impulse can be calculated in proportion to \( r \), which is the distance travelled by the sound wave until it reaches \( L \). The drop in sound pressure is proportional to \( 1/r \). The number of reflections should also be taken into account, since obstacles usually do not reflect the whole energy of the incoming wave. The loss of energy due to a reflection can be calculated based on the reflective property of the material of the geometric objects. In our implementation we use a reflection coefficient specifying how much of the energy is reflected.

Reflections also alter the frequency content of the reflected wave. Rough materials do not reflect the energy in the high frequency components of a signal as good as the low frequency components, leading to a low pass filtering effect. This effect can be modelled by placing low-pass filtered versions of an impulse into the frequency response instead of using impulses. Gauss-curves of a specific width can be used for this purpose.

Using a negative reflection coefficient for each reflection lets us calculate an impulse response whose amplitude is centered around zero, which is usually the case in recorded room impulse responses, as was also remarked by Lehmann and Johansson [11]. When placing the impulses or Gauss curves into the impulse response waveform, the sign on the placed impulse or curve is calculated as \( (-1)^{N_R} \), where \( N_R \) is the number of times the wave has been reflected until it reaches point \( L \).

6. Impulse response based on early reflections

Calculating the impulse response representing the early reflections this way has a major disadvantage. When the listener moves from point \( L \) to a point \( L' \), the distances between the listener’s position and the virtual sound sources change and so do the sample numbers at which the impulses or curves for each reflection are placed. When the player moves around slowly, the moving impulses in the impulse response inflict an easily noticeable comb filter like quality onto the signal. This artefact can be seen in figure 5, which shows the spectrogram of white noise to which impulse responses based on the different listener positions have been applied. The changing resonant frequencies can be seen clearly. This effect is much stronger than...
in reality, because the number of reflections considered in our simulation is only a fraction of the number of reflections found in reality. This discrepancy is due to the neglecting of important effects such as diffraction and dispersion and the simulation in two dimensional space instead of three dimensional space.

Figure 5. Impulse responses of early reflections calculated for 4 different positions have been calculated using only impulses to model the reflections. The resonance in thin frequency bands caused by these impulse responses is clearly visible on the white noise the impulse responses have been applied to.

To mitigate this effect, short bursts of white noise can be placed into the signal instead of using impulses or gauss curves. We use noise bursts whose length is proportional to the number of reflections the wave has undergone until it reaches point $L$. An envelope in the shape of a gauss curve is applied to the noise bursts. These noise bursts can be interpreted as dispersed versions of reflections. To simulate the low-pass filtering characteristics of the reflecting material, a low-pass filter whose characteristics depend on the number of reflections can be applied to the noise used in the bursts.

Whenever the position of the listener is updated, the impulse response needs to be recalculated. If the listener’s position changes only slightly or gradually over time, which is often the case when a player walks from one point to another in a video game, the impulse response should also change gradually. To fulfill this condition also when using noise bursts for all reflections, the same noise samples need to be used for the same reflection every time. This can be achieved by reading noise from a noise buffer at a random position that is constant for each reflection or by providing a seed specific to each reflection to the random number generator before drawing the noise samples.

During the calculation of the circular sectors, the circle was divided into a certain number of sectors right from the beginning. When neighbouring beams hit the same obstacle, they are reflected in the same way, i.e. their underlying circles will have the same center point. Such neighbouring sectors need to be treated as the same reflection using the same noise samples, otherwise the player will hear a transition when moving from the area of one sector to the area of another sector. Two sectors $i$ and $j$ are considered as being neighbours when $P_i = P_j$, $N_{R_i} = N_{R_j}$ and either $\phi_1_i = \phi_2_j$ or $\phi_2_i = \phi_1_j$, where the subscripts mark the properties of the respective sectors.

7. Calculation of late reflections
The late reflections are estimated based on the early reflections and on parameters that can be controlled by the sound designer, since the desired amount of reverberation is often a question of game design rather than fidelity to physics.

Late reflections are modelled by white noise with an exponentially decaying amplitude. The amplitude of the noise is based on the amplitude of the later part of the impulse response of the early reflections. The mean amplitude is calculated as follows:

$$A = \frac{\sum_{n=N_h-k}^{N_h} |h[n]|}{k \cdot M}$$  \hspace{1cm} (1)
where $h[n]$ is the impulse response containing the early reflections, $N_h$ the number of samples in $h[n]$, $k$ a constant that we define as $k = \lfloor N_h/4 \rfloor$ and $M$, used to normalize the average, is the mean of the absolute samples of white noise with amplitude 1.

The exponential decay $\lambda$ can be calculated by calculating as

$$\lambda = -\frac{\log(A/A_0)}{3k}$$

where $A_0$ is the amplitude of the first $k$ samples of the impulse response, calculated in a similar way as the amplitude $A$ as

$$A_0 = \frac{\sum_{n=0}^{k} |h[n]|}{k \cdot M}.$$  

The decay value can then be used to calculate the amplitude at sample $t$ as

$$A[t] = A_0 \cdot e^{-\lambda t}. $$

In our implementation the decay is often set to a custom value by a sound designer overriding the calculated value in favour of sound design aspects over fidelity to physical relations.

The samples of the white noise used to create the late reflections are the same each time the impulse response of the late reflections is recreated. Also here the reason is that this helps us to create a consistent sound that won’t change unnaturally whenever the late reflections are re-estimated. A low pass filter can be applied to the noise being used to account for the fact that higher frequency components are often dampened in reflected sounds waves.

The decaying noise is added to $h$ to yield a complete impulse response, representing the direct sound, the early reflections and the late reflections.

8. Convolution and implementation details

The reverberation represented by the impulse response $h$ is applied to the sound signal of the sound source by convolving the sound with the impulse response $h$. Executing the convolution in the frequency domain is very efficient. If the convolution and the FFT necessary to transform the signal to the frequency domain are implemented on a GPU, considerable gains in speed can be expected.

Convolving part of a signal of $M$ samples length with an impulse response of $N$ samples length will yield a convolution product of $L = M + N - 1$ samples length and hence requires a Fourier transform of at least $L$ samples length to accommodate the whole product. Since typical impulse responses for large spaces are a few seconds long, this would mean that blocks of $L$ samples, where $L$ is a number corresponding to a few seconds, need to be processed. This would also increase the latency of the system to a few seconds. One alternative is to fill buffers of $L$ samples with only $M$ samples of the signal and transform them to the frequency domain for convolution, keeping the latency caused by the convolution as low as $M$ samples, but increasing the computational cost significantly by using FFTs with $L$ coefficients while only processing $M$ samples.

This problem can be solved by using partitioned fast convolution [16], in which the impulse response is cut into short frames of $M$ samples length and each frame is transformed into the frequency domain with a $2M$ order FFT. These frames are stored in the array $H[t]$, where $t = 0...T - 1$ is the index of the frame and $T = \lceil L/M \rceil$. To process a frame of $M$ samples of the input signal at frame number $n$, it is also transformed to the frequency domain with a $2M$ order FFT, yielding frame $S[n]$. The output frame at frame number $n$ is calculated as

$$X[n] = \sum_{t=0}^{T-1} H[t] \cdot S[n - t],$$
where the dot stands for the sample-wise complex multiplication of two frames. To implement this formula, \( T - 1 \) frames of the past signal have to be kept in memory but for every frame of \( M \) samples, only one FFT and one iFFT of order \( 2M \) are necessary.

Figure 6. Summary of the partitioned convolution algorithm with overlapping input and output frames.

When the impulse response \( h \) is updated frequently, as is the case when the player moves through the level, the timings where the early reflections are placed in the impulse response change frequently. This can cause artefacts in the output because succeeding frames of the output signal are convolved with different impulse responses. An easy solution to this problem is to process overlapping buffers to which an appropriate windowing function is applied before transforming them into the frequency domain. We use a periodic Hann window and a 50% overlap. The relation between the frame number \( n \) and its first sample \( i \) in the time domain is \( i = n \times M/2 \), so the frame \( S[n] \) is constructed by taking \( M \) samples starting from sample \( i \), then multiplying it with the Hann window, padding it with zeros to \( 2M \) samples and transforming it to the frequency domain. Note that the output \( X[n] \) of the convolution needs to be overlapped with the same overlap, which means that the relation between frame number and sample number in the time domain is the same. Figure 6 summarizes the convolution algorithm graphically.

To generate multichannel output the impulse response must be constructed as a multi channel signal. Each channel of the impulse response is then individually applied to the mono input signal.

9. Summary and outlook

The proposed method is a fast way to implement reverberation for virtual environments. Even though diffraction and dispersion are neglected during pre-computation of the early reflections, stochastic enhancement of the calculated impulse response lets us create convincingly sounding acoustic effects. The method works well for environments that can be easily converted to 2d maps but needs to be extended for environments with a more complicated geometry. Further optimization of the image source pre-calculations would enable us to also implement this part of the algorithm in real-time and thus allow us to calculate the sound for moving sound sources.

References

[1] Lauri Savioja and U Peter Svensson, “Overview of geometrical room acoustic modeling techniques,” The Journal of the Acoustical Society of America, vol. 138, no. 2, pp. 708–730, 2015.
[2] Nikunj Raghuvanshi and John Snyder, “Parametric wave field coding for precomputed sound propagation,” ACM Transactions on Graphics (TOG), vol. 33, no. 4, pp. 38, 2014.
[3] Carl Schissler and Dinesh Manocha, “Interactive sound propagation and rendering for large multi-source scenes,” ACM Transactions on Graphics (TOG), vol. 36, no. 1, pp. 2, 2016.
[4] Dirk Schröder, Philipp Dross, and Michael Vorländer, “A fast reverberation estimator for virtual environments,” in Audio Engineering Society Conference: 30th International Conference: Intelligent Audio Environments. Audio Engineering Society, 2007.
[5] Fabio Antonacci, Marco Foco, Augusto Sarti, and Stefano Tubaro, “Fast tracing of acoustic beams and paths through visibility lookup,” IEEE Transactions on Audio, Speech, and Language Processing, vol. 16, no. 4, pp. 812–824, 2008.
[6] Eva-Marie Nosal, Murray Hodgson, and Ian Ashdown, “Improved algorithms and methods for room sound-field prediction by acoustical radiosity in arbitrary polyhedral rooms,” *The Journal of the Acoustical Society of America*, vol. 116, no. 2, pp. 970–980, 2004.

[7] Lakulish Antani, Anish Chandak, Micah Taylor, and Dinesh Manocha, “Direct-to-indirect acoustic radiance transfer,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 261–269, 2012.

[8] Jont B Allen and David A Berkley, “Image method for efficiently simulating small-room acoustics,” *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[9] Jeffrey Borish, “Extension of the image model to arbitrary polyhedra,” *The Journal of the Acoustical Society of America*, vol. 75, no. 6, pp. 1827–1836, 1984.

[10] FP Mechel, “Improved mirror source method in roomacoustics,” *Journal of sound and vibration*, vol. 256, no. 5, pp. 873–940, 2002.

[11] Eric A Lehmann and Anders M Johansson, “Prediction of energy decay in room impulse responses simulated with an image-source model,” *The Journal of the Acoustical Society of America*, vol. 124, no. 1, pp. 269–277, 2008.

[12] Hansung Kim, Richard J Hughes, Luca Remaggi, Philip JB Jackson, Adrian Hilton, Trevor J Cox, and Ben Shirley, “Acoustic room modelling using a spherical camera for reverberant spatial audio objects,” in *Audio Engineering Society Convention 142*. Audio Engineering Society, 2017.

[13] Uwe M Stephenson, “Quantized pyramidal beam tracing-a new algorithm for room acoustics and noise immission prognosis,” *Acta Acustica united with Acustica*, vol. 82, no. 3, pp. 517–525, 1996.

[14] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West, “A beam tracing approach to acoustic modeling for interactive virtual environments,” in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 21–32.

[15] Vesa Valimaki, Julian D Parker, Lauri Savioja, Julius O Smith, and Jonathan S Abel, “Fifty years of artificial reverberation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[16] Barry D Kulp, “Digital equalization using fourier transform techniques,” in *Audio Engineering Society Convention 85*. Audio Engineering Society, 1988.