Room Impulse Response Generator

dr.ir. Emanuël A.P. Habets
ehabets@dereverberation.org

September 20, 2010

Abstract

This report provides a short overview of different methods that can be used for simulating room acoustics and focuses on the imagemethod that was proposed by Allen and Berkley in 1979. The image method is probably one of the methods most commonly used in the acoustic signal processing community, and will therefore be discussed in more detail. A mex-function, which can be used in MATLAB, has been created to generate multichannel Room Impulse Responses using the image method. This function enables the user to control the reflection order, room dimension and microphone directivity.

1 Introduction

Many people who are working in the field of acoustic signal processing reach a point where they want to simulate room acoustics. This report gives a short overview of different methods that can be used for simulating room acoustics. The imagemethod [1], which was proposed by Allen and Berkley in 1979, is probably one of the methods most commonly used in the acoustic signal processing community. This method is closely related to the so-called wave equation and its frequency domain counterpart called the Helmholtz equation. These equations describe the propagation of acoustic waves through fluids (gas or liquid).

A mex-function, which can be used in MATLAB, has been created to generate multi-channel Room Impulse Responses (RIR) using the image method. This function enables the user to control the reflection order, room dimension and microphone directivity. Another advantage of this mex-function, compared to a standard MATLAB function, concerns the computation time. We have found that our mex-function was around 100 times faster than the standard MATLAB code on our test machine.

This report is organized as follows. The wave equation and the Helmholtz equation are discussed in Sections 2 and 3 respectively. In Section 4 a short overview of different methods that can be used to simulate room acoustics is provided. Allen and Berkley’s method is described in Section 5. A short description of the mex-function implementation is provided in Section 6. Some examples are presented in Section 7. Finally, a short summary is provided in Section 8. Please note and respect the copyrights in Section 11.

2 Wave Equation

In principle, any complex sound field can be considered as a superposition of numerous simple sound waves (e.g., plane waves), and their propagation within a room can be con-
Considered linear if the properties of the medium in which the waves travel are assumed to be homogeneous, at rest, and independent of wave amplitude [2]. In physics, the wave equation governs the propagation of waves through fluids (gas or liquid). The form of the equation is a second order partial differential equation. The equation describes the evolution of velocity potential or sound pressure \( p(\mathbf{r},t) \) as a function of position \( \mathbf{r} = [x,y,z] \) and time \( t \).

For a homogeneous medium undergoing inviscid fluid flow, one can linearize the equations governing the dynamic behaviour of the fluid, namely the Euler’s equation (i.e., Newton’s 2nd law applied to fluids), the continuity equation, and the state equation, to obtain the wave equation

\[
\nabla^2 p(\mathbf{r},t) - \frac{1}{c^2} \frac{\partial^2 p(\mathbf{r},t)}{\partial t^2} = 0, \tag{1}
\]

where

\[
\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{2}
\]

is the Laplacian expressed in the Cartesian coordinates \((x,y,z)\), and \(c\) is the speed of sound. The wave equation accurately describes the pressure in the sound field provided \(|p(\mathbf{r},t)| \ll \rho_0 c^2\), where \(\rho_0\) is the density of the propagation medium at equilibrium.

In practice, two types of inhomogeneities occur in the medium: i) scalar inhomogeneities (spatial distribution of sound speed and air density), e.g., due to temperature variations in the medium, and ii) vector inhomogeneities (spatial distribution of particle mean velocity), e.g., due to the presence of fans or an air conditioning. However, the effects of these inhomogeneities are so small that they can often be ignored in room acoustics.

In order to calculate the sound field emanating from a source in a specific room we need an additional source function in (1) and boundary conditions that describe the sound reflection and absorption at the walls. Let \(s(\mathbf{r},t)\) denote the source function, then the wave equation is given by

\[
\nabla^2 p(\mathbf{r},t) - \frac{1}{c^2} \frac{\partial^2 p(\mathbf{r},t)}{\partial t^2} = -s(\mathbf{r},t). \tag{3}
\]

3 Helmholtz Equation

Let us consider the wave equation in the frequency domain. The Fourier transform is defined as

\[
P(\mathbf{r};\omega) \triangleq \mathcal{F}\{p(\mathbf{r},t)\}(\omega) = \int_{-\infty}^{\infty} p(\mathbf{r},t) \exp(i\omega t) \, dt, \tag{4}\]

where \(\exp(x) = e^x\) and \(i = \sqrt{-1}\). By applying the Fourier transform to (1) the time-independent Helmholtz equation is obtained, i.e.,

\[
\nabla^2 P(\mathbf{r};\omega) + k^2 P(\mathbf{r};\omega) = 0, \tag{5}\]

where \(k\) denotes the wave number that is related to the angular frequency \(\omega\) and the wave length \(\lambda\) through

\[
k = \frac{\omega}{c} = \frac{2\pi}{\lambda}. \]

If there is a harmonic disturbance which is producing the waves, for which the source function is given by \(s(\mathbf{r},t) = S(\mathbf{r};\omega)e^{-i\omega t}\), then the Helmholtz equation is given by

\[
\nabla^2 P(\mathbf{r};\omega) + k^2 P(\mathbf{r};\omega) = -S(\mathbf{r};\omega). \tag{6}\]

For a unit-amplitude harmonic point source at position \(\mathbf{r}_s = [x_s, y_s, z_s]\) we have \(S(\mathbf{r};\omega) = \delta(\mathbf{r} - \mathbf{r}_s) = \delta(x-x_s)\delta(y-y_s)\delta(z-z_s)\), where \(\delta(\cdot)\) denotes the Kronecker delta function. The
partial differential equation in (6) can be solved by first solving the following inhomogeneous equation:

\[ \nabla^2 H(r, r_s; \omega) + k^2 H(r, r_s; \omega) = -\delta(r - r_s), \tag{7} \]

where \( H(r, r_s; \omega) \) is the Room Transfer Function (RTF), or Green’s function. For an arbitrary source function \( S(r_s; \omega) \) the desired source pressure can then be calculated using the following relation

\[ P(r; \omega) = \iiint_{V_s} H(r, r_s; \omega) S(r_s; \omega) \, dr_s, \tag{8} \]

where \( V_s \) denotes the source volume, and \( dr_s = dx_s \, dy_s \, dz_s \) is the differential volume element at position \( r_s \). The sound pressure \( p(r, t) \) can now be obtained using the inverse Fourier transform of (8).

The conventional way to solve (7) is to find an orthogonal set of eigenfunctions associated with the Laplacian operator and then to expand \( H(r, r_s; \omega) \) as a sum of eigenfunctions. Specifically, a function \( \Psi_m(r; \omega) \) that satisfies the homogenous equation \((\nabla^2 + k^2)\Psi_m(r; \omega) = 0\) over a certain interval and satisfies certain boundary conditions at the end of the interval, is called an eigenfunction. Subsequently, a general expression for the Green’s function \( H(r, r_s; \omega) \) in an arbitrary sound field can be obtained using the eigenfunctions:

\[ H(r, r_s; \omega) = \sum_{m=0}^{\infty} C_m(r_s; \omega) \Psi_m(r; \omega), \tag{9} \]

where each coefficient \( C_m(r_s; \omega) \) depends on the position of the sound source. The eigenfunctions depend on the boundary conditions imposed by the enclosed space.

**Free Space Green’s Function**

For an omnidirectional point source in an unbounded space, i.e., free space, the Green’s function is \[2\]

\[ H(r, r_s; \omega) = \frac{\exp(i \omega \|r - r_s\|)}{4\pi \|r - r_s\|}, \tag{10} \]

where \( \| \cdot \| \) denotes the Euclidean norm.

**Classical Rectangular Room**

In a rectangular room with physical dimensions \((L_x, L_y, L_z)\) and rigid perfectly reflecting walls the eigenfunctions in Cartesian coordinates are \[2\]

\[ \Psi_m(r) = \cos \left( \frac{m_x \pi}{L_x} x \right) \cos \left( \frac{m_y \pi}{L_y} y \right) \cos \left( \frac{m_z \pi}{L_z} z \right), \tag{11} \]

where \( m = (m_x, m_y, m_z) \) and \( m_v \in \{x, y, z\} \) are nonnegative integers. Let us define \( k_v = m_v \pi / L_v \) for \( v \in \{x, y, z\} \), then we can write (11) as

\[ \Psi_m(r) = \cos(k_x x) \cos(k_y y) \cos(k_z z). \tag{12} \]

The eigenfunctions are often referred to as modes and have a simple physical interpretation as three-dimensional standing waves. The corresponding eigenvalues are \( k_m^2 = k_x^2 + k_y^2 + k_z^2 \).

The solution for the inhomogeneous equation (7) for a classical rectangular room is \[2\]

\[ H(r, r_s; \omega) = \sum_{m \in M} \frac{\Psi_m(r) \Psi_m^*(r_s)}{\Lambda_m(k_m^2 - k_m^2)}, \tag{13} \]
where $\mathcal{M} = \{(m_x, m_y, m_z) : m_x, m_y, m_z \in \mathbb{N}_0\}$ denotes a set that contains all desired triples $m$ and $\Lambda_m$ is a normalization constant for the associated eigenvector defined by

$$\int \int \int_{\mathcal{V}} \Psi_m(r) \Psi_m^*(r) \, dr = \begin{cases} \Lambda_m, & \text{for } m = n; \\ 0, & \text{otherwise}, \end{cases}$$

(14)

where $\mathcal{V} = \{(x, y, z) : 0 \leq x \leq L_x, 0 \leq y \leq L_y, 0 \leq z \leq L_z\}$ is the entire space of the room and $dr = dx \, dy \, dz$ is the differential volume element at position $r$.

Equation (13) reveals the frequency domain structure of the RTF. The eigenfrequencies $\omega_m$, related to the eigenvalues through $k_m = \frac{\omega_m}{c}$, are also known as the resonance frequencies of the room. At each eigenfrequency $\omega_m$, the standing wave pattern of mode $m$ resonates strongly. From (13) it can be seen that $H(r, r_s; \omega)$ increases without bound as $\omega \to \omega_m$. Room mode $\Psi_m(r)$ is said to be excited at eigenfrequency $\omega_m$ (i.e., $\Psi_m(r)$ makes a large contribution to sound pressure at this frequency). All rooms possess distinct resonances at low frequencies. However, in practical rooms, where walls are non-rigid and finitely absorbing, eigenvalues $k_m$ have imaginary components that provide damping of resonance modes [2]. In that case $k_m = \frac{\omega_m}{c} + i \frac{\delta_m}{c}$, where $\delta_m$ denotes the damping constant (Q-factor).

Assuming that $\delta_m \ll \omega_m$, (13) results in

$$H(r, r_s; \omega) = \frac{c^2}{\sum_{m \in \mathcal{M}} \frac{\Psi_m(r) \Psi_m^*(r_s)}{\Lambda_m(\omega^2 - \omega_m^2 - 2i\delta_m\omega_m)}}.$$  

(15)

The inverse Fourier transform of the frequency response of the room described by (13) leads to a RIR, $h(r, r_s, t)$. The form of (13) justifies the use of some well-known modelling techniques used in signal processing such as, for example, the pole-zero model.

4 Simulating Room Acoustics

Mathematically the sound propagation is described by the wave equation. An impulse response from a source to a microphone can be obtained by solving the wave equation. Since it can seldom be expressed in an analytic form the solution must be approximated. There are three main modelling methods, as illustrated in Figure 1, viz., wave-based, ray-based and statistical [3]. The ray-based methods, such as the ray-tracing [4] and the image-source method [1], are the most often used. The wave-based methods, such as the Finite
Element Method (FEM), Boundary Element Method (BEM) [5, 6] and Finite-Difference Time-Domain (FTDT) [7] methods, are computational more demanding. In real-time auralization\(^1\) the limited computation capacity requires simplifications. A frequently used simplification consists of modelling the direct path and early reflections individually and the late reflections by recursive digital filter structures. The statistical modelling methods, such as the Statistical Energy Analysis, have been widely used in aerospace, ship and automotive industry for high frequency noise analysis and acoustic designs. They are not suitable for auralization purposes since those methods do not model the temporal behaviour of a sound field.

Wave-based methods

The most accurate results can be achieved by wave-based methods. An analytical solution for the wave equation can be found only in extremely simple cases such as a rectangular room with rigid walls. Therefore, numerical methods such as FEM and BEM [5, 6] are often used. The main difference between these two element methods is in the element structure. In FEM, the space is divided into volume elements, while in BEM only the boundaries of the space are divided into surface elements. The elements interact with each other according to the basics of wave propagation. The sizes of these elements have to be much smaller than the size of the wavelength for every particular frequency. At high frequencies, the required number of elements becomes very high, resulting in a large computational complexity. Therefore, these methods are suitable only for low frequencies and small enclosures.

Another method for room acoustics simulation is provided by the FDTD method [7, 8]. The main principle of this method is that derivatives in the wave equation are replaced by corresponding finite differences. The FDTD method produces impulse responses that are better suited for auralization than FEM and BEM. The main benefit of the element methods over FDTD methods is that one can create a denser mesh structure where required, such as locations near corners or other acoustically challenging places.

In all wave-based methods, the most difficult part is the definition of the boundary conditions and geometrical description of the objects. Typically a complex impedance is required, but it is hard to find that data in existing literature.

Ray-based methods

The ray-based methods are based on geometrical room acoustics [2]. The most commonly used ray-based methods are the ray-tracing [4] and the image method [1]. The main difference between these methods is the way the reflection paths are calculated [3]. To model an ideal impulse response from a source to a receiver all possible sound reflection paths, commonly called rays, should be discovered. In ray-tracing methods the sound power emitted by a sound source is described by a finite number of rays. These rays propagate through space and are reflected after every collision with the room boundaries. During that time, their energy decreases as a consequence of the sound absorption of the air and of the walls involved in the propagation path. When the rays reach the receiver, an energy calculation process is performed. When all rays are processed the impulse response is obtained. Rays can be selected from a set of randomly distributed angles, uniformly distributed angles or from a restricted set of angles. Due to this the ray-tracing methods are by no means exhaustive, whereas the image method finds all the rays. However, while the image method is limited to geometries that are formed by planer surfaces the ray-tracing method can be applied to geometries that are formed by arbitrary surfaces.

\(^1\)Auralization is the process of rendering audible, by physical or mathematical modelling, the sound field of a source in a space, in such a way as to simulate the binaural listening experience at a given position in the modelled space.
It should be mentioned that all ray-based methods are based on energy propagations. This means that all effects involving phase differences such as refraction or interference are neglected. This is admissible if the sound signals of interest are not sinusoids or other signals with small frequency bandwidth but are composed of many spectral components covering a wide frequency range. Then it can be assumed that constructive and destructive phase effects cancel each other when two or more sound field components superimpose at a point, and the total energy in the considered point is simply obtained by adding their energies. Components with this property are often referred to as mutually incoherent [9].

5 Allen and Berkley’s Image Method

The image model can be used to simulate the reverberation in a room for a given source and microphone location, and is discussed in Section 5.1. Using the image method Allen and Berkley [1] developed an efficient method to compute a Finite Impulse Response (FIR) that models the acoustic channel between a source and a receiver in rectangular rooms. The image method and some additional refinements will we discussed in Section 5.2. The close relation of the image method and the Helmholtz equation is shown in Section 5.3.

5.1 Image Model

Figure 2 shows a sound source $S$ located near a rigid reflecting wall. At destination $D$ two signals arrive, one from the direct path and a second one from the reflection. The path length of the direct path can be directly calculated from the known locations of the source and the destination. Also shown is an image of the source, $S'$, located behind the wall at a distance equal to the distance of the source from the wall. Because of symmetry, the triangle $SRS'$ is isosceles and therefore the path length $SR + RD$ is the same as $S'D$. Hence, to compute the path length of the reflected path, we can construct an image of the source and compute the distance between destination and image. Also, the fact that we are computing the distance using one image means that there was one reflection in the path.

Figure 3 shows a path involving two reflections. The length of this path can be obtained from the length of $S''D$. In Figure 4 the length of a path involving three reflections is obtained from the length of $S'''D$. These figures can also be extended to three dimensions to take into account reflections from the ceiling and the floor.

In general the path lengths (and thus the delays) of reflections can be obtained by computing the distance between the source images and the destination. The strength of the reflection can be obtained from the path length and the number of reflections involved in the path. The number of reflections involved in the path is equal to the level of images that was used to compute the path.
Figure 3: Path involving two reflections obtained using two images.

Figure 4: Path involving three reflections obtained using three images.
5.2 Image Method

Consider a rectangular room with length, width and height given by \( L_x, L_y \) and \( L_z \). Let the sound source be at a location represented by the vector \( \mathbf{r}_s = [x_s, y_s, z_s] \) and let the microphone be at a location represented by the vector \( \mathbf{r} = [x, y, z] \). Both vectors are with respect to the origin, which is located at one of the corners of the room. The relative positions of the images measured with respect to the receiver position and obtained using the walls at \( x = 0, y = 0 \) and \( z = 0 \) can be written as

\[
\mathbf{R}_p = [(1 - 2q)x_s - x, (1 - 2j)y_s - y, (1 - 2k)z_s - z].
\]

Each of the elements in the triple \( p = (q, j, k) \) can take on values 0 or 1, resulting in eight different combinations that specify a set \( \mathcal{P} \), i.e., \( \mathcal{P} = \{(q, j, k) : q, j, k \in \{0, 1\}\} \). When the value of \( p \) is 1 in any dimension, then an image of the source in that direction is considered. It should be noted that some of these images correspond to higher order reflections. To consider all images, we add the vector \( \mathbf{R}_m \) to \( \mathbf{R}_p \) where

\[
\mathbf{R}_m = [2m_x L_x, 2m_y L_y, 2m_z L_z],
\]

where \( m_x, m_y \), and \( m_z \) are integer values. Each of the elements of the triple \( m = (m_x, m_y, m_z) \) takes on values from \(-N\) to \(+N\). The reflection order related to an image at the position \( \mathbf{R}_p + \mathbf{R}_m + \mathbf{r} \) is given by

\[
O_{p,m} = |2m_x - q| + |2m_y - j| + |2m_z - k|.
\]

The distance between any source image and the microphone can be written as

\[
d = \|\mathbf{R}_p + \mathbf{R}_m\|.
\]

The time delay of arrival of the reflected sound ray corresponding to any source image can be expressed as

\[
\tau = \frac{d}{c} = \frac{\|\mathbf{R}_p + \mathbf{R}_m\|}{c},
\]

where \( c \) denotes the sound velocity in meters per second.

The impulse response for this source and microphone location can now be written as

\[
h(\mathbf{r}, \mathbf{r}_s, t) = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \beta_{x_1}^{m_{x_1}-q} \beta_{x_2}^{m_{x_2}} \beta_{y_1}^{m_{y_1}-j} \beta_{y_2}^{m_{y_2}} \beta_{z_1}^{m_{z_1}-k} \beta_{z_2}^{m_{z_2}} \frac{\delta(t - \tau)}{4\pi d},
\]

where \( \mathcal{M} = \{(m_x, m_y, m_z) : -N \leq m_x, m_y, m_z \leq N\} \) denotes a set that contains all desired triples \( m \). The quantities \( \beta_{x_1}, \beta_{x_2}, \beta_{y_1}, \beta_{y_2}, \beta_{z_1} \) and \( \beta_{z_2} \) are the reflection coefficients of the six walls. Note that the walls at \( v = 0 \) with \( v \in \{x, y, z\} \) correspond to \( \beta_{v_1} \), and that the walls at \( v = L_v \) with \( v \in \{x, y, z\} \) correspond to \( \beta_{v_2} \). The elements of the triple \( p \) are 0 or 1, which means that there are 8 different combinations, \((0,0,0)\) to \((1,1,1)\). The elements of the triple \( m \) range from \(-N\) to \(+N\), which means that there are \((2N+1)^3\) combinations. Therefore, for a given \( N \), this method computes \(8(2N+1)^3\) different paths. The delays of the impulses corresponding to these paths are computed using (20) and the strengths of these impulses are multiplied by reflection coefficients as many times as there are reflections. Once the impulse response has been computed this way, the source signal can be convolved with the impulse response to simulate the signal picked up by the microphone.

An important consideration while simulating the discrete version of this impulse response using a computer is that the delays given by (20) do not always fall at sampling instants. Ideally, the discrete version of (21) is given by

\[
h(\mathbf{r}, \mathbf{r}_s, n) = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \beta_{x_1}^{m_{x_1}-q} \beta_{x_2}^{m_{x_2}} \beta_{y_1}^{m_{y_1}-j} \beta_{y_2}^{m_{y_2}} \beta_{z_1}^{m_{z_1}-k} \beta_{z_2}^{m_{z_2}} \frac{\text{LPF}\{\delta(n - \tau f_s)\}}{4\pi d},
\]

where \( \text{LPF} \) is a low-pass filter function.
where $f_s$ is the sampling frequency and LPF{·} denotes a theoretically perfect Low-Pass Filter with cut-off frequency $f_s/2$. In [1] the time-of-arrival (in samples) was shifted to the nearest integer value. Hence, the following approximation was made

$$\text{LPF}\{\delta(n - \tau f_s)\} \approx \delta(n - \text{round}(\tau f_s)).$$

(23)

Although this distortion can be ignored in many applications, for multiple microphone systems that are sensitive to inter-microphone phase, correct simulation of arrival time relationships is critical. One way to reduce this problem is to compute the discrete impulse response at a much higher sampling frequency, decimate the impulse response to the original sampling frequency, and convolve the source signal with it. Peterson suggested another modification to the image method [10]. In this approach, each impulse in (21) is replaced by the impulse response of a Hanning-windowed ideal low-pass filter of the form

$$\delta_{\text{LPF}}(t) = \frac{1}{2} \left( 1 + \cos \left( \frac{2\pi}{T_w} t \right) \right) \text{sinc} \left( \frac{2\pi}{f_c} t \right)$$

for $-\frac{T_w}{2} < t < \frac{T_w}{2}$, otherwise

(24)

where $T_w$ is the width (in time) of the impulse response and $f_c$ is the cut-off frequency of the low-pass filter. For the simulations performed in this report $T_w$ was set to 4 ms and $f_c$ was set to the Nyquist frequency. Each impulse $\delta(t - \tau)$ in (21) is first replaced by $\delta_{\text{LPF}}(t - \tau)$ and subsequently sampled. By doing this, true delays of arrival of the reflected signals are simulated accurately even at the original low sampling frequency. A comparison of both methods is depicted in Figure 5, where the delay was set to 4.8 samples. Squares indicate sample values produced by Allen and Berkley’s shifted impulse method and circles indicate values produced by Peterson’s low-pass impulse method. The solid line shows the central portion of the continuous-time low-pass impulse function.

The other consideration while simulating reverberation for a room is the duration of reverberation or the reverberation time. Formally, the reverberation time is defined as the time required for the intensities of reflected sound rays to be down 60 dB from the direct path sound ray. An empirical formula, known as Sabin-Franklin’s formula [11] can be used to relate the reverberation time $RT_{60}$ by,

$$RT_{60} = \frac{24 \ln(10) V}{c \sum_{i=1}^{6} S_i (1 - \beta_i^2)}.$$ 

(25)

where $V$ denotes the volume of the room, and $\beta_i$ and $S_i$ denote the reflection coefficient and the surface of the $i^{th}$ wall, respectively.

Figure 5: Comparison of the shifted and low-pass impulse method.
5.3 Relation with the Helmholtz Equation

The image model described in Section 5.1 yields an intuitive explanation of the image-method. In this section we will proof the relation between the solution of the Helmholtz equation (7) for a classical rectangular room and (21). The solution of the Helmholtz equation for a rectangular room was given in (13).

Let us take all images into account by defining $-\infty < m_v < \infty$ for $v \in \{x,y,z\}$. By expanding the cosines of the eigenfunctions using exponentials, and by using the fact that $\Lambda_m = V \forall m$, we can write (13) as

$$H(r_1, r_2; \omega) = \frac{1}{8V} \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \exp(\frac{ik_m \cdot R_p}{k^2 - \|k_m\|^2}),$$

where

$$k_m = \left[ \frac{m_x \pi}{L_x}, \frac{m_y \pi}{L_y}, \frac{m_z \pi}{L_z} \right] = [k_x, k_y, k_z],$$

and $R_p$ represents the eight vectors given by (16).

Now we proceed along the same lines as Allen and Berkley in [1]. Using the property of the delta function on $k_x, k_y,$ and $k_z$, i.e.

$$\int_{-\infty}^{\infty} \delta(k - a) f(k) = f(a),$$

we may rewrite (26) in integral form

$$H(r_1, r_2; \omega) = \frac{1}{8V} \sum_{p \in \mathcal{P}} \int\int\int_{-\infty}^{\infty} \exp(\frac{ik' \cdot R_p}{k^2 - \|k\|^2}) \sum_{m \in \mathcal{M}} \delta(k' - k_m) \, dk',$$

where $k' = [k'_x, k'_y, k'_z]$. By Fourier series analysis one may show

$$\sum_{m_v = -\infty}^{\infty} \delta \left( k'_v - \frac{m_v \pi}{L_v} \right) = \frac{L_v}{\pi} \sum_{m_v = -\infty}^{\infty} \exp(i2L_v m_v k'_v) \text{ for } v \in \{x,y,z\}.\quad(30)$$

Therefore, we can rewrite (29) as

$$H(r_1, r_2; \omega) = \frac{1}{(2\pi)^3} \sum_{p \in \mathcal{P}} \int\int\int_{-\infty}^{\infty} \sum_{m \in \mathcal{M}} \exp(\frac{ik' \cdot (R_p + R_m)}{k^2 - \|k\|^2}) \, dk',$$

where $R_m$ is the vector define in (17). The integral in (31) is just a plane wave expansion for a point source in free space since

$$\frac{\exp(ik\|R\|)}{4\pi\|R\|} = \frac{1}{(2\pi)^3} \int\int\int_{-\infty}^{\infty} \exp(\frac{ik' \cdot R}{k^2 - \|k\|^2}) \, dk'.\quad(32)$$

Finally, using (31) and (32) we obtain

$$H(r_1, r_2; \omega) = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \frac{\exp(ik\|R_p + R_m\|)}{4\pi\|R_p + R_m\|}.$$\quad(33)

Taking the inverse Fourier transform\(^2\) of (33) the RIR is obtained

$$h(r_1, r_2, t) = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \frac{\delta(t - \|R_p + R_m\|/c)}{4\pi\|R_p + R_m\|}\quad(34)$$

$$= \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \frac{\delta(t - \tau)}{4\pi d}.\quad(35)$$

\(^2\)Note that the inverse Fourier transform is defined as $p(t) = \int_{-\infty}^{\infty} P(\omega) \exp(-i\omega t) \, d\omega$. 
By taking into account the reflection coefficients of the six walls we obtain

\[
h(r, r_s, t) = \sum_{p \in P} \sum_{m \in \mathcal{M}} \beta_{\|m_x - q\|} \beta_{\|m_y - r\|} \beta_{\|m_y - r\|} \beta_{\|m_z - k\|} \frac{\delta(t - \tau)}{4\pi d},
\]

which is equivalent to (21).

\section{Implementation}

The image method as discussed in the previous section has been implemented as a MATLAB mex-function and was written in C++. The resulting Dynamic-Link-Library (DLL) can easily be used within MATLAB as a standard MATLAB function. The C++ implementation is much faster than the equivalent MATLAB implementation. The source-code can be found in Appendix A.

The function \textit{rir\_generator} is defined as follows:

\begin{verbatim}
function [h, beta_hat] = rir_generator(c, fs, r, s, L, beta, nsample, 
mtype, order, dim, orientation, 
hp_filter);
\end{verbatim}

**Input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>sound velocity in m/s.</td>
</tr>
<tr>
<td>fs</td>
<td>sampling frequency in Hz.</td>
</tr>
<tr>
<td>r</td>
<td>M x 3 matrix specifying the (x,y,z) coordinates of the receiver(s) in m.</td>
</tr>
<tr>
<td>s</td>
<td>1 x 3 vector specifying the (x,y,z) coordinates of the source in m.</td>
</tr>
<tr>
<td>L</td>
<td>1 x 3 vector specifying the room dimensions (x,y,z) in m.</td>
</tr>
<tr>
<td>beta</td>
<td>1 x 6 vector specifying the reflection coefficients (\beta) or beta = Reverberation Time (RT\textsubscript{60}) in seconds.</td>
</tr>
</tbody>
</table>

**Optional input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsample</td>
<td>number of samples to calculate.</td>
</tr>
<tr>
<td>mtype</td>
<td>type of microphone that is used ['omnidirectional', 'subcardioid', 'cardioid', 'hypercardioid', 'bidirectional'].</td>
</tr>
<tr>
<td>order</td>
<td>maximum reflection order.</td>
</tr>
<tr>
<td>dim</td>
<td>room dimension (2 or 3).</td>
</tr>
<tr>
<td>orientation</td>
<td>direction in which the microphone is pointed, specified using azimuth and elevation angles in radians.</td>
</tr>
<tr>
<td>hp_filter</td>
<td>use 'false' to disable high-pass filter.</td>
</tr>
</tbody>
</table>

**Default value**

<table>
<thead>
<tr>
<th>nsample</th>
<th>RT\textsubscript{60}/fs</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtype</td>
<td>'omnidirectional'</td>
</tr>
<tr>
<td>order</td>
<td>-1</td>
</tr>
<tr>
<td>dim</td>
<td>3</td>
</tr>
<tr>
<td>orientation</td>
<td>[0 0]</td>
</tr>
<tr>
<td>hp_filter</td>
<td>'true'</td>
</tr>
</tbody>
</table>

**Output parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>M x nsample matrix containing the calculated room impulse response(s).</td>
</tr>
<tr>
<td>beta_hat</td>
<td>In case a reverberation time is specified as an input parameter the corresponding reflection coefficient is returned.</td>
</tr>
</tbody>
</table>

**Multi-Channel Support** In case more than one receiver position is specified the function \textit{rir\_generator} will calculate all RIRs at once.

**Reverberation Time versus Reflection Coefficients** The reflection coefficients in (21) can be specified using the parameter 'beta'. In case 'beta' consists of one element the
program assumes a reverberation time (in seconds) is specified. The corresponding average reflection coefficient is calculated using (25) and will be returned using the output parameter ‘\( \beta_{\text{hat}} \).’

**Reflection Order and Room Dimension** In order to control the complexity of the generated RIR one can control the maximum reflection order using the parameter ‘\( \text{order} \).’ In case the order is chosen ‘-1’ (default value) the maximum amount of reflections, given the desired length of the RIR, is calculated. The dimension of the room can be set using the parameter ‘\( \text{dim} \).’ This value can either be 2 or 3 (default value).

**Microphone Directivity** The microphone’s directionality, or polar pattern, can also be taken into account. Different kinds of polar patterns are implemented and can be chosen using the parameter ‘\( \text{mtype} \).’ The signal attenuation \( A(\theta) \), where \( \theta \) denotes the direction of arrival, is calculated using the following standard formula:

\[
A(\theta) = \alpha + (1 - \alpha) \cos(\theta).
\]  

(37)

The polar pattern is controlled by \( \alpha \), see Table 1. The resulting polar patterns for the Omnidirectional, Cardioid, Hypercardioid and Bidirectional microphone are depicted in Figure 6.

<table>
<thead>
<tr>
<th>Directivity Pattern</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omnidirectional (Monopole)</td>
<td>1</td>
</tr>
<tr>
<td>Subcardioid</td>
<td>0.75</td>
</tr>
<tr>
<td>Cardioid</td>
<td>0.5</td>
</tr>
<tr>
<td>Hypercardioid</td>
<td>0.25</td>
</tr>
<tr>
<td>Bidirectional (Dipole)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Supported polar patterns and corresponding values for \( \alpha \).

The angle in which the microphone is pointing can be adjusted with the parameter ‘\( \text{orientation} \).’ By default the microphone points towards the positive x-axis. The microphone’s directionality takes the azimuth and elevation of the received reflections into account.
7 Examples

In this section some basic and more complex examples are presented in the form of a MATLAB® script.

\[ c = 340 \text{ m/s} \]
\[ f_s = 16000 \text{ samples/s} \]
\[ r = [2 \ 1.5 \ 2] \text{ m} \]
\[ s = [2 \ 3.5 \ 2] \text{ m} \]
\[ L = [5 \ 4 \ 6] \text{ m} \]
\[ \beta = 0.4 \]
\[ n = 4096 \]
\[ h = \text{rir \_generator}(c, f_s, r, s, L, \beta, n) \]

Example 1: Simple example to generate one RIR.

Figure 7: Output of Example A.1.
% Sound velocity (m/s)  
c = 340;

% Sample frequency (samples/s)  
fs = 16000;

% Receiver position [x y z] (m)  
r = [2 1.5 2];

% Source position [x y z] (m)  
s = [2 3.5 2];

% Room dimensions [x y z] (m)  
L = [5 4 6];

% Reverberation time (s)  
beta = 0.4;

% Number of samples  
n = 1024;

% Type of microphone  
mtyp = 'omnidirectional';

% Reflection order  
order = 2;

% Room dimension  
dim = 3;

% Microphone orientation (rad)  
orientation = [0 0];

% Enable high-pass filter  
hp_filter = 1;

h = rir_generator(c, fs, r, s, L, beta, n, mtype, order, dim, orientation, hp_filter);

Example 2: Generate one RIR.

![](image.png)

Figure 8: Output of Example A.2.

% Sound velocity (m/s)  
c = 340;

% Sample frequency (samples/s)  
fs = 16000;

% Receiver position [x y z] (m)  
r = [2 1.5 2; 1 1.5 2];

% Source position [x y z] (m)  
s = [2 3.5 2];

% Room dimensions [x y z] (m)  
L = [5 4 6];

% Reverberation time (s)  
beta = 0.4;

% Number of samples  
n = 4096;

% Type of microphone  
mtyp = 'omnidirectional';

% Reflection order  
order = -1;

% Room dimension  
dim = 3;

% Microphone orientation (rad)  
orientation = [pi/2 0];

% Enable high-pass filter  
hp_filter = 1;

h = rir_generator(c, fs, r, s, L, beta, n, mtype, order, dim, orientation, hp_filter);

Example 3: Generate multiple RIRs.

% Sound velocity (m/s)  
c = 340;

% Sample frequency (samples/s)  
fs = 16000;

% Receiver position [x y z] (m)  
r = [2 3.5 2];

% Source position [x y z] (m)  
s = [2 3.5 2];

% Room dimensions [x y z] (m)  
L = [5 4 6];

% Reverberation time (s)  
beta = 0.4;

% Number of samples  
n = 4096;

% Type of microphone  
mtyp = 'hypercardioid';

% Reflection order  
order = -1;

% Room dimension  
dim = 3;

% Microphone orientation (rad)  
orientation = [pi/2 0];

% Enable high-pass filter  
hp_filter = 1;

h = rir_generator(c, fs, r, s, L, beta, n, mtype, order, dim, orientation, hp_filter);

Example 4: Generate one RIR using a hypercardioid microphone.
A reverberant signal can now be created by filtering the anechoic signal with the generated RIR as shown in Example 5.

\texttt{reverberant\_signal = \texttt{fft\_filt}(h, clean\_signal);}  

Example 5: Generate reverberant signal.

\section{Summary}

In this report we presented a short overview of different methods for simulating room acoustics. We discussed the well-known image method, proposed by Allen and Berkley \cite{1}, in more detail. An efficient implementation of the image method was developed in the form of a MATLAB mex-function written in C++. Some example scripts were presented to demonstrate the use of this function. Our implementation incorporates some novel features which allows the user to control the complexity of the RIR and the directivity pattern of the receiver.

\section{Acknowledgements}

The author expresses his thanks to J. van de Laar and N.D. Gaubitch for proofreading the initial version of this manuscript and testing the mex-function, and to D. Jarrett for fruitful discussions and comments that gave rise to the latest version of the RIR Generator and this tutorial.

\section{Contact Information}

If you have any comments and/or suggestions you can contact the author at the following address:

\textbf{Mail:} d.i.r. Emamiël A.P. Habets  
\hspace{1cm} Mailstop: EEE-CSP  
\hspace{1cm} Department of Electrical and Electronic Engineering  
\hspace{1cm} Imperial College London  
\hspace{1cm} Exhibition Road  
\hspace{1cm} London SW7 2AZ  
\hspace{1cm} United Kingdom

\textbf{E-mail:} ehabets@dereverberation.org
11 Copyrights


This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

A Source Code

/*
 * Program: Room Impulse Response Generator
 * Description: Computes the response of an acoustic source to one or more microphones in a reverberant room using the image method [1, 2].
 * Author: dr. ir. E. A. P. Habets (ehabets@derverberation.org)
 * Version: 2.0.20100920
 * History:
 * 1.0.20030606 Initial version
 * 1.1.20040901 + Microphone directivity
 * 1.2.20040312 + Reflection order
 * 1.3.20050930 + Reverberation Time
 * 1.4.20051114 + Supports multi-channels
 * 1.5.20051116 + High-pass filter [2]
 * 1.6.20060327 + Microphone directivity control
 * 1.7.20060511 + Minor improvements
 * 1.8.20060713 + Minor improvements
 * 1.9.20090602 + 3D microphone directivity control
 * 2.0.20100920 + Calculation of the source-image position changed in the code and tutorial.
 * This ensures a proper response to reflections in case a directional microphone is used.
 */

#define USE_MATH_DEFINES
#include "matrix.h"
#include "mex.h"
#include "math.h"
#define ROUND(x) ((x)>0?(long)((x)+0.5):(long)((x)-0.5))
#endif
#define M_PI 3.14159265358979323846
#endif

double sinc(double x) {

```c
if (x == 0)
    return(1);
else
    return(sin(x)/x);
}

double sim_microphone(double x, double y, double z, double strength, char mtype)
{
    switch(mtype)
    {
    case 'b':
        alpha = 0;
        break;
    case 'h':
        alpha = 0.25;
        break;
    case 'c':
        alpha = 0.5;
        break;
    case 's':
        alpha = 0.75;
        break;
    }
    vartheta = acos(z/sqrt(pow(x,2)+pow(y,2)+pow(z,2)));
    varphi = atan2(y,x);
    strength = sin(M_PI/2-angle[1]) + sin(vartheta) + cos(angle[0]-varphi);
    strength = alpha + (1-alpha)*strength;
    return strength;
}
```
impulse

Please change the arguments!

Hat = 0;

be arguments!

Hat = filter;
filtration input

The arguments.

many 2003 Netherlands.

required.

are

current arguments.

least Emanuel Habets

Generator

Hat = mxGetPr(plhs[1]);

doubles beta_hat = mxGetPr(prhs[1]);

doubles beta_hat[0] = 0;

// Reflection coefficients or Reverberation Time?

if (mxGetN(prhs[5]) == 1) {


TR = beta_hat[0];

double alfa = 24 * V * log (10.0) / (c * S * TR);

if (alfa > 1)

mexErrMsgTxt("Error: The reflection coefficients cannot be calculated using the current -

room parameters \( V \) or \( S \) or reverberation time \( TR \)\). Please specify the reflection coefficients or change the room parameters.");

beta_hat[0] = sqrt((1 - alfa);

for (int i = 0; i < 6; i++)

beta[i] = beta_hat[0];

}

else {

for (int i = 0; i < 6; i++)

beta[i] = beta_hat[i];

}

// High-pass filter (optional)

if (nrhs > 11 && mxIsEmpty(prhs[11]) == false) {

hp_filter = (int) mxGetScalar(prhs[11]);

}

else {

hp_filter = 1;

}

// 3D Microphone orientation (optional)

if (nrhs > 10 && mxIsEmpty(prhs[10]) == false) {

const doubles orientation = mxGetPr(prhs[10]);

if (mxGetN(prhs[10]) == 1) {

angle[0] = orientation[0];

angle[1] = 0;

} else {

angle[0] = orientation[0];

angle[1] = orientation[1];

}

else {

angle[0] = 0;

angle[1] = 0;

}

// Room Dimension (optional)

if (nrhs > 9 && mxIsEmpty(prhs[9]) == false) {

}
dim = (int) mxGetScalar(prhs[9]);
if (dim != 2 || dim != 3)
    mexErrMsgTxt("Invalid input arguments!");
if (dim == 2)
    { beta[4] = 0;
      beta[5] = 0; }
else
    { dim = 3; }
// Reflection order (optional)
if (nrhs > 8 && mxIsEmpty(prhs[8]) == false)
    { order = (int) mxGetScalar(prhs[8]);
    if (order < -1)
        mexErrMsgTxt("Invalid input arguments!");
    } else
    { order = -1; }
// Type of microphone (optional)
if (nrhs > 7 && mxIsEmpty(prhs[7]) == false)
    { mtype = new char[mxGetN(prhs[7]) + 1];
    mxGetString(prhs[7], mtype, mxGetN(prhs[7]) + 1);
    } else
    { mtype = new char[1];
      mtype[0] = 'o'; }
// Number of samples (optional)
if (nrhs > 6 && mxIsEmpty(prhs[6]) == false)
    { nsamples = (int) mxGetScalar(prhs[6]);
    } else
    { if (mxGetN(prhs[5]) > 1)
        { double V = LL[0] * LL[1] * LL[2];
          double alpha = ((1 - pow(beta[0], 2)) + (1 - pow(beta[1], 2)) * LL[0] * LL[2] +
                          ((1 - pow(beta[4], 2)) + (1 - pow(beta[5], 2)) * LL[0] * LL[1]);
          TR = 24 * log(10.0) * V / (c * alpha);
          if (TR < 0.128)
              TR = 0.128;
        }
        nsamples = (int) (TR * fs);
    }
// Create output vector
plhs[0] = mxCreateDoubleMatrix(nrofmics, nsamples, mxREAL);
double* imp = mxGetPr(plhs[0]);
// Temporary variables and constants (high-pass filter)
const double W = 2 * M_PI * 100 / fs;
const double R1 = exp(-W);
const double B1 = 2 * R1 * cos(W);
const double B2 = -R1;
const double A1 = -(1 + R1);
double X0;
double* Y = new double[3];
// Temporary variables and constants (image-method)
const double Fs = 1;
const int Tw = 2 * ROUND(0.004 * fs);
const double CTs = c / Fs;
double* hanning_window = new double[Tw + 1];
double* L1 = new double[Tw + 1];
double* r = new double[3];
double* L = new double[3];
double hu[6];
double* s = new double[3];
double* L = new double[3];
double hu[6];
double ref[3];
double dist;
double ll;
double strength;
int poa, fdist;
int m1, n2, n3;
int qj, k;
int mx, my, nz;
s[0] = ss[0] / CTs;
s[1] = ss[1] / CTs;
L[0] = LL[0] / CTs;
// Hanning window
for (n = 0; n < Tw + 1; n++)
    hanning_window[n] = 0.5 * (1 + cos(2 * M_PI * (n * Tw / 2) / Tw));
}
for (int mic_nr = 0; mic_nr < nrofmics; mic_nr++)
    { // s[0], s[1], ... s[N], s[0], s[1], ... s[N], s[0], s[1], ... s[N] }
\[ r[0] = r[\text{mic}nr + 0*nrof\text{mics}]/\text{cTs}; \]
\[ r[1] = r[\text{mic}nr + 1*nrof\text{mics}]/\text{cTs}; \]
\[ r[2] = r[\text{mic}nr + 2*nrof\text{mics}]/\text{cTs}; \]
\[ n1 = (\text{int}) \left\lceil \frac{\text{nsamples}/(2*L[0])}{cTs} \right\rceil; \]
\[ n2 = (\text{int}) \left\lceil \frac{\text{nsamples}/(2*L[1])}{cTs} \right\rceil; \]
\[ n3 = (\text{int}) \left\lceil \frac{\text{nsamples}/(2*L[2])}{cTs} \right\rceil; \]

// Generate room impulse response
for (mx = -n1; mx <= n1; mx++) {
    \[ hu[0] = 2*mx*L[0]; \]
    for (my = -n2; my <= n2; my++) {
        \[ hu[1] = 2*my*L[1]; \]
        for (mz = -n3; mz <= n3; mz++) {
            \[ hu[2] = 2*mz*L[2]; \]
            for (q = 0; q < 1; q++) {
                \[ hu[3] = (1-2*q)*s[0] - r[0] + hu[0]; \]
                \[ refl[0] = \text{pow}(\beta[0], \text{abs}(mx-q)); \]
                for (j = 0; j < 1; j++) {
                    \[ refl[1] = \text{pow}(\beta[1], \text{abs}(my-j)); \]
                    for (k = 0; k < 1; k++) {
                        \[ refl[2] = \text{pow}(\beta[2], \text{abs}(mz-k)); \]
                        \[ \text{dist} = \sqrt{\text{pow}(hu[3], 2) + \text{pow}(hu[4], 2) + \text{pow}(hu[5], 2)}; \]
                        if (abs(2*mx-q)+abs(2*my-j)+abs(2*mz-k) < order || order == -1) {
                            \[ fdist = (\text{int}) \left\lfloor \text{dist} \right\rfloor; \]
                            \[ \text{strength} = \text{sim}\text{-microphone}(hu[3], hu[4], hu[5], \text{angle}, \text{mtype}[0]) * \text{refl}[0] * \text{refl}[1] * \text{refl}[2] / (4*\text{M}\text{PI}\text{dist} \times \text{cTs}); \]
                            for (n = 0; n < \text{Tw}+1; n++) {
                                \[ \text{LPI}[n] = \text{hanning}\text{-window}[n] * Fc * \text{sinc}(\text{M}\text{PI}\text{Fd} \times (n - \text{dist} - \text{fdist}) - (\text{Tw}/2)); \]
                                \[ \text{imp}[\text{mic}nr+nrof\text{mics} \times (n+fdist)] += \text{strength} \times \text{LPI}[n]; \]
                            }
                        }
                        \[ \text{pos} = \text{fdist} - (\text{Tw}/2); \]
                        for (n = 0; n < \text{Tw}+1; n++) {
                            \[ \text{imp}[\text{mic}nr+nrof\text{mics} \times (n+\text{pos})] += \text{strength} \times \text{LPI}[n]; \]
                        }
                    }
                }
            }
        }
    }
}

// 'Original' high-pass filter as proposed by Allen and Berkley.
if (\text{hp_filter} == 1) {
    \[ \text{Y}[0] = 0; \]
    for (int idx = 0; idx < 3; idx++) \{ Y[idx] = 0; \}
    for (int idx = 0; idx < \text{nsamples}; idx++) \{
        X0 = \text{imp}[\text{mic}nr+nrof\text{mics} \times idx];
        \[ Y[2] = Y[1]; \]
        \[ Y[1] = Y[0]; \]
        \[ Y[0] = \text{B1}Y[1] + \text{B2}Y[2] + X0; \]
        \[ \text{imp}[\text{mic}nr+nrof\text{mics} \times idx] += Y[0] * \text{A1} + Y[1] * \text{B1} + Y[2]; \]
    }
}

References


