

Fast image method for impulse response calculations of box-shaped rooms

Stephen G. McGovern

Abstract

The image method has been used for calculating the impulse response of box-shaped rooms for over 25 years. While this method is functional, it can be inefficient because many of the commonly used mathematical operations are either redundant or unnecessary. This paper addresses these two inefficiencies by proposing both the use of look-up tables to prevent redundant calculations, and the use of a sorting method to allow the prevention of unnecessary calculations. Either technique, by itself, results in a large reduction in computation time. The greatest time reductions, however, can be achieved when both techniques are used together.

Key words: room impulse response; auralisation; auralization; reverberation; room acoustics

PACS: 43.55.Ka

1 Introduction

Computer modeling of room acoustics dates as far back as 1962[1]. In the early 1970's Gibbs and Jones[2] used the image method to digitally model room acoustics, and in 1978 Allen and Berkley[3] used the image method to perform impulse response calculations for box-shaped rooms with specular and angle independent reflections. Later, methods were developed that accounted for complexities such as arbitrarily shaped rooms[4] and diffuse reflections[5]. Another method accounting for these complexities is ray tracing[6], and a method called finite element analysis models room acoustics with very good accuracy[7]. Room acoustics have also been modeled with methods that originated in the field of computer graphics[8], and the general rendering of acoustic space has been improved upon by the implementation of models for both directional sources and receivers[9] and head-related shadowing effects[10].

Many of the methods used to model complex acoustic environments require significant computation time. This can become very burdensome when

Email address: `stech@2pi.us` (Stephen G. McGovern).

URL: `www.2pi.us` (Stephen G. McGovern).

many thousands of impulse responses are needed. When shorter computation time takes precedence over the complexity of the acoustic environment, then impulse response calculations might have greater practicality if they are intended to simulate simple environments. This could potentially be the case for experiments involving time dependent room impulse responses[11], blind source separation[12], and dereverberation[13].

Computation time has been topic of many papers and there have been many efforts to reduce it. Gains in efficiency have been made by applying techniques that include extrapolation methods [14], fast beam tracing methods [15], binary space partitioning [16], and fast multipole methods [17]. Each of these methods, however, has some kind of drawback. Extrapolation generates an approximation, while fast beam tracing and binary space partitioning are intended for complex room shapes, and the fast multipole method ceases to be fast if there are multiple locations for both the receiver and the source.

The remainder of this paper presents an optimized image method for calculating the impulse response of a box-shaped room. The optimization results from a reduction in the number of necessary mathematical operations, and the output is theoretically identical to that produced by Allen and Berkley.

In Sec. 2 a discussion regarding two approaches to optimization is presented. These approaches are elaborated on in the two following sections. Specifically, there is a discussion about the creation of look-up tables, followed by a discussion regarding their optimized implementation. The next section contains both a discussion of the algorithm's performance and a comparison of the algorithm's output to that produced by the algorithm proposed by Allen and Berkley. The paper then ends with a few concluding remarks. An implementation of the algorithm, written in the C programming language, has also been included as an appendix to the paper.

2 Optimizations

2.1 Look-Up Tables

The impulse response for a box shaped room can be given by

$$h(t) = \sum_{i=-N_x}^{N_x} \sum_{j=-N_y}^{N_y} \sum_{k=-N_z}^{N_z} p(t)_{ijk} \quad \text{for } 0 \leq t \leq t_{length}, \quad (1)$$

where t_{length} is the length of the impulse response and $p(t)_{ijk}$ is the pressure resulting from the ijk^{th} image. The triple summation given in Eq. 1 sums over a total of N_{total} terms, where

$$N_{total} = (2N_x + 1)(2N_y + 1)(2N_z + 1). \quad (2)$$

If the distance from the ijk^{th} image to the receiver is given by x_i , y_j , and z_k , then the square of these distances is given by $(x_i)^2$, $(y_j)^2$, $(z_k)^2$. Find-

ing these squared values for each of the N_{total} terms requires a significant number of calculations. Finding the values of the corresponding reflection coefficients, which are given by a_i , b_j , and c_k , also requires a significant number of calculations. This is especially true for higher order reflections because the calculation of the coefficient for each n^{th} order reflection requires a minimum of n multiplies.

The number of calculations is reduced by finding the values of $(x_i)^2$, $(y_j)^2$, $(z_k)^2$, a_i , b_j , and c_k prior to the summation and placing the values in look-up tables. The pre-computed values are then accessed as needed during the summation process. While some time is spent on the pre-computation, this part of the procedure itself is very cheap, and each value is reused many time during the summation.

In this paper, look-up tables are denoted by bold fonts. For the look-up tables corresponding to the square of the distances, this is $(\mathbf{x}^2)_i$, $(\mathbf{y}^2)_j$, and $(\mathbf{z}^2)_k$. For the look-up tables corresponding to the reflection coefficients, this is \mathbf{a}_i , \mathbf{b}_j , and \mathbf{c}_k . The range of index i for the look-up tables is shifted from $-N_x \leq i \leq N_x$ to $0 \leq i \leq 2N_x$. Similarly, the shift in range for the j and k indices is to $0 \leq j \leq 2N_y$ and $0 \leq k \leq 2N_z$ respectively. For the impulse response, this yields the equation

$$h(t) = \sum_{i=0}^{2N_x} \sum_{j=0}^{2N_y} \sum_{k=0}^{2N_z} p(t)_{ijk}. \quad (3)$$

This change is not necessary, but it avoids the awkwardness of having look-up tables with negative indices. The practicality of this increases when sorting methods are used. Details regarding the creation of look-up tables are given in Sec. 3

2.2 Sorted Look-Up Tables

The impulse response in Eq. 3 can be used only if it has a finite time length. This time length can be chosen arbitrarily, but one may want to take various factors into account. If the chosen time length is multiplied by the speed of sound, a distance is obtained. This distance can be thought of as a radius that forms a boundary around the receiver. In three dimensions the region inside of this boundary is a sphere. In two dimensions it is a circle, and in one dimension it is a line. The second scenario is illustrated in Fig. 1.

Despite the fact that the three-dimensional summation takes place over a cube of lattice points, only the lattice points falling within the spherical boundary contribute to the impulse response. Image points falling outside of the boundary are unwanted and information about them is ordinarily discarded. The unwanted image points, however, occur only for particular combinations of i , j , and k . Further optimization is achieved by first predicting at what combinations of indices unwanted image points will occur, and then skipping over those sections of the lattice during the summation process. The

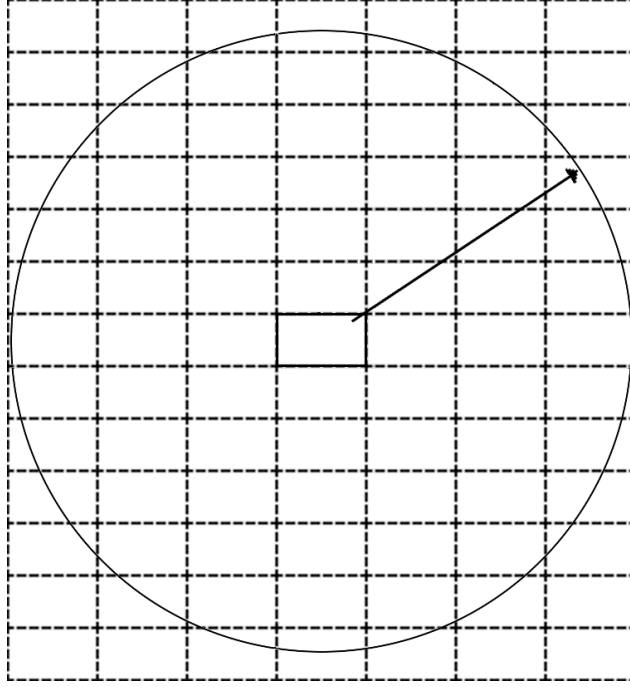


Fig. 1. Diagram of a room lattice showing the distance boundary. Only virtual sources located inside of the circle contribute to the impulse response.

reduction in the number of mathematical operation can be roughly approximated by finding the percent difference between the volume of a sphere and the volume of a cube. Assuming that the diameter of the sphere is equal to the width of the cube, the percent difference works out to

$$\left(\frac{V_{\text{cube}} - V_{\text{sphere}}}{V_{\text{cube}}} \right) 100\% = 47.64\%. \quad (4)$$

In practice, the actual percent difference is almost always higher than this. This occurs because the lattice only approximates the shape of a sphere. If the number of echoes within the spherical boundary is only 200, then the savings in the number of distance calculations can be in excess of 60%.

To aid in the prediction of unwanted images, the values held in the square-of-the-distance look-up tables are sorted so as to have ascending values. This property is described by the relations

$$(\mathbf{x}^2)_{i+1} \geq (\mathbf{x}^2)_i \quad (5)$$

$$(\mathbf{y}^2)_{j+1} \geq (\mathbf{y}^2)_j \quad (6)$$

$$(\mathbf{z}^2)_{k+1} \geq (\mathbf{z}^2)_k, \quad (7)$$

where the i , j , and k^{th} elements of the reflection coefficient tables follow a corresponding pattern. A description explaining how this property leads to the exclusion of unwanted images is provided in Sec. 4.

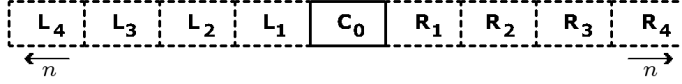


Fig. 2. One-dimensional diagram of room images. Numbers on the diagram denote order of reflection, and letters denote whether the reflection is to the left or right of the real room.

Table 1

In one dimension there are a total of four possible echo patterns. The terms C_0 , L_n , R_n , and $(\mathbf{x}^2)_i$ represent the sound from the real source, the sound from the n^{th} order image on the left side, the sound from the n^{th} order image from the right side, and the corresponding i^{th} term of the square-of-the-distance look-up table respectively.

direct sound	odd order		even order		odd order		even order	
$(\mathbf{x}^2)_0$	$(\mathbf{x}^2)_1$	$(\mathbf{x}^2)_2$	$(\mathbf{x}^2)_3$	$(\mathbf{x}^2)_4$	$(\mathbf{x}^2)_5$	$(\mathbf{x}^2)_6$	$(\mathbf{x}^2)_7$	$(\mathbf{x}^2)_8$
C_0	L_1	R_1	L_2	R_2	L_3	R_3	L_4	R_4
C_0	L_1	R_1	R_2	L_2	L_3	R_3	R_4	L_4
C_0	R_1	L_1	R_2	L_2	R_3	L_3	R_4	L_4
C_0	R_1	L_1	L_2	R_2	R_3	L_3	L_4	R_4

3 Creation of Sorted Look-Up Tables

3.1 Echo Arrival Patterns in One Dimension

Consider the model in one dimension as shown in Fig. 2. The real room is labeled with a C_0 . In this case, the "C" is intended to indicate that the room is at the center of the image lattice and the "0" is intended to indicate that it corresponds a 0^{th} order reflection. The phrase "zeroth order reflection" simply means that the sound has undergone zero reflections. To the right of C_0 , there are four rooms labeled as R_1 , R_2 , R_3 , and R_4 . The R indicates that the images are to the right of the center image and the subscripts 1, 2, 3, and 4 indicate that they correspond to the 1^{st} , 2^{nd} , 3^{rd} , and 4^{th} order reflections respectively. An arbitrary n^{th} order image on the right side can be referred to by the term R_n . The same scheme also exists on the left side, only in this case the rooms are labeled L_1 , L_2 , L_3 , and L_4 . Similarly, an arbitrary n^{th} order image on the left side can be referred to by the term L_n .

When the sound source in the room creates a noise, a series of echoes is produced. These echoes arrive from the virtual sources at the receiver in a particular pattern. Whatever this pattern is, it will be consistent with one of four possibilities. These possibilities are given in Table 1.

Inspection of Table 1 reveals that the direct sound is always first. Closer inspection reveals that echoes following the direct sound arrive in pairs. Within a pair, one echo is from the left, one echo is from the right, and both echoes are of the same order of reflection. It can also be observed that in each possible

Table 2

The pattern of individual echoes in a pair is determined by these conditions. The pattern for echo pairs of odd order are determined by the conditions in the column on the far left, and the pattern for echo pairs of even order are determined by the conditions in the column second from the left. The pattern itself is given in the two right most columns.

condition for odd ordered pairs	condition for even ordered pairs	former arrival	latter arrival
$x_r + x_s \leq 0$	$x_r - x_s \leq 0$	L_n	R_n
$x_r + x_s > 0$	$x_r - x_s > 0$	R_n	L_n

pattern the lower ordered pairs always arrive prior to the higher ordered pairs.

Rather than determining which of the four patterns is followed, a pattern of either $[L_n, R_n]$ or $[R_n, L_n]$ can be found for the individual echoes within each pair. This determination can be made by applying one of two rule sets. One set of rules is applied to echo pairs of odd order and the other is applied to echo pairs of even order. The odd order pair and the even order pair rules are:

For Odd Order Pairs: If the sum of the receiver distance and the source distance is less than zero, the left echo comes first. Otherwise, if the sum is greater than zero, then the right echo comes first. If the sum is equal to zero, then the echoes occur at the same time and either order is valid.

For Even Order Pairs: If the receiver distance subtracted by the source distance is less than zero, then the left echo comes first. If the receiver distance subtracted by the source distance is greater than zero, then the right echo comes first. If the receiver distance and the source distance are equal, then the echoes occur at the same time and either order is valid.

In the cases where either order is valid, a convention that presumes that the left echo arrives first will be used. This convention is observed throughout this paper.

A summary of the two rule sets is found in Table 2 where the location of the receiver, the location of the source, and the width of the room are given by the variables x_r , x_s , and x_w respectively. These variables are depicted in a diagram of the real room in Fig. 3

3.2 Mathematics of Sorted Look-Up Tables

Before finding the i^{th} element of the square-of-the-distances look-up table, the distance from the receiver to either the C_0 , the L_n , or the R_n image must be known. The distances from the receiver to the C_0 , the L_n , and the R_n images are given by Eqs. 8, 9, 10 respectively.

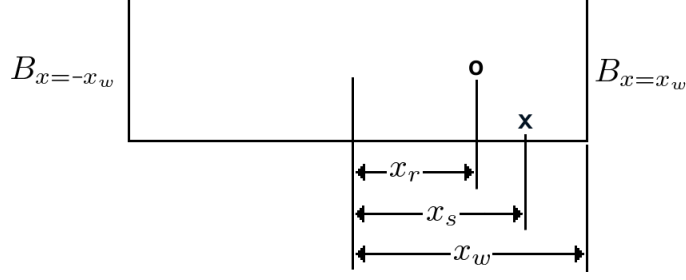


Fig. 3. The real room with measurements along the x -axis. The center of the room is taken to be the origin, x_s is the location of the source relative to the origin, x_r is the location of the receiver relative to origin, and x_w is the distance from the center of the room to the wall. The variables $B_{x=-x_w}$ and $B_{x=x_w}$ represent the reflection coefficients for the two walls perpendicular to the x -axis.

$$x_{C_0} = x_s - x_r \quad (8)$$

$$x_{L_n} = \begin{cases} x_{L_{n-1}} - 2(x_w + x_s) & \text{for odd } n \\ x_{L_{n-1}} - 2(x_w - x_s) & \text{for even } n \end{cases} \quad (9)$$

$$x_{R_n} = \begin{cases} x_{R_{n-1}} + 2(x_w - x_s) & \text{for odd } n \\ x_{R_{n-1}} + 2(x_w + x_s) & \text{for even } n \end{cases} \quad (10)$$

The values of x_{L_0} and x_{R_0} are both found by Eq. 11.

$$x_{L_0} = x_{R_0} = x_{C_0} \quad (11)$$

Using Eq. 8, the zeroth element in the square-of-the-distance look-up table is found by equation Eq. 12. From Eqs. 9 and 10, the distances of the former and latter members of each echo pair are found. These correspond to odd and even elements of the square-of-the-distance look-up table and are given by Eqs. 13 and 14 respectively.

$$(\mathbf{x}^2)_0 = (x_{C_0})^2 \quad (12)$$

$$(\mathbf{x}^2)_{2i-1} = \begin{cases} (x_{L_{n=i}})^2 & \text{for odd } i \text{ and } x_r + x_s \leq 0 \\ (x_{L_{n=i}})^2 & \text{for even } i \text{ and } x_r - x_s \leq 0 \\ (x_{R_{n=i}})^2 & \text{for odd } i \text{ and } x_r + x_s > 0 \\ (x_{R_{n=i}})^2 & \text{for even } i \text{ and } x_r - x_s > 0 \end{cases} \quad (13)$$

$$(\mathbf{x}^2)_{2i} = \begin{cases} (x_{R_{n=i}})^2 & \text{for odd } i \text{ and } x_r + x_s \leq 0 \\ (x_{R_{n=i}})^2 & \text{for even } i \text{ and } x_r - x_s \leq 0 \\ (x_{L_{n=i}})^2 & \text{for odd } i \text{ and } x_r + x_s > 0 \\ (x_{L_{n=i}})^2 & \text{for even } i \text{ and } x_r - x_s > 0 \end{cases} \quad (14)$$

Just as with the square-of-the-distance look-up table, the reflection coefficient for either the C_0 , the L_n , or the R_n image must be known before the i^{th} value of the reflection coefficient table can be found. Referring again to Fig. 3, the reflection coefficient for the wall perpendicular to the x -axis and to the left of the origin is given by $B_{x=-x_w}$. The reflection coefficient for the wall opposite that is given by $B_{x=x_w}$. Using these variables the reflection coefficients for the C_0 , the L_n , and the R_n images are given by Eqs. 15, 16, and 17 respectively.

$$a_{C_0} = 1 \quad (15)$$

$$a_{L_n} = \begin{cases} B_{x=-x_w} a_{L_{n-1}} & \text{for odd } n \\ B_{x=x_w} a_{L_{n-1}} & \text{for even } n \end{cases} \quad (16)$$

$$a_{R_n} = \begin{cases} B_{x=x_w} a_{R_{n-1}} & \text{for odd } n \\ B_{x=-x_w} a_{R_{n-1}} & \text{for even } n \end{cases} \quad (17)$$

Where

$$a_{L_0} = a_{R_0} = a_{C_0} \quad (18)$$

From Eq. 15 the reflection coefficient for the C_0 image is given by Eq. 19. From Eqs. 16 and 17 the reflection coefficients for the former and latter elements of each echo pair are given by Eqs. 20, and 21 respectively.

$$\mathbf{a}_0 = a_{C_0} \quad (19)$$

$$\mathbf{a}_{2i-1} = \begin{cases} a_{L_{n=i}} & \text{for odd } i \text{ and } x_r + x_s \leq 0 \\ a_{L_{n=i}} & \text{for even } i \text{ and } x_r - x_s \leq 0 \\ a_{R_{n=i}} & \text{for odd } i \text{ and } x_r + x_s > 0 \\ a_{R_{n=i}} & \text{for even } i \text{ and } x_r - x_s > 0 \end{cases} \quad (20)$$

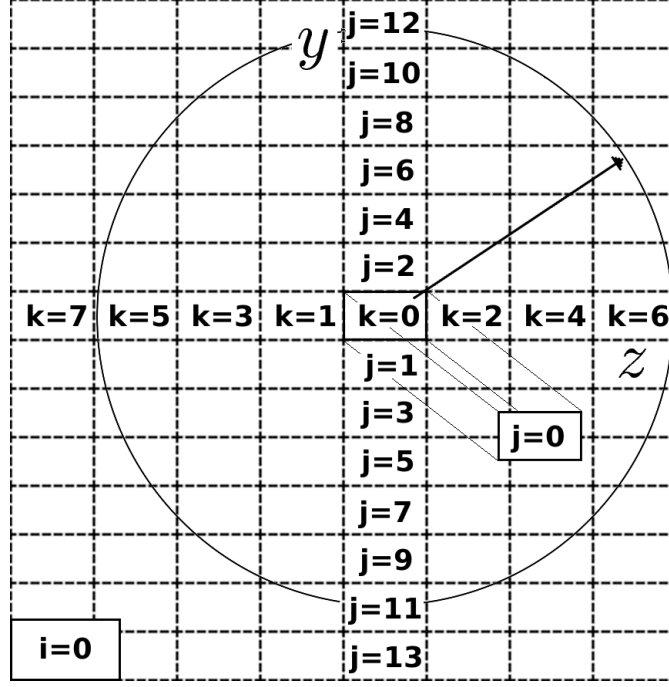


Fig. 4. A two-dimensional diagram of a room image lattice. Lower values of indices k and j lie toward the center of the circle. These values increase as the distance from the center is increased.

$$\mathbf{a}_{2i} = \begin{cases} a_{R_{n=i}} & \text{for odd } i \text{ and } x_r + x_s \leq 0 \\ a_{R_{n=i}} & \text{for even } i \text{ and } x_r - x_s \leq 0 \\ a_{L_{n=i}} & \text{for odd } i \text{ and } x_r + x_s > 0 \\ a_{L_{n=i}} & \text{for even } i \text{ and } x_r - x_s > 0 \end{cases} \quad (21)$$

4 Implementation of Sorted Look-Up Tables

Using the illustration of the image lattice in Fig. 1, the indices j and k , for the sorted look-up tables, can be mapped to their corresponding room images. This is shown in figure Fig. 4 where it is assumed that the images along both the y and the z axes follow a pattern of the form

$$L_{\text{odd}}, R_{\text{odd}}, L_{\text{even}}, R_{\text{even}}, \dots$$

When the summation in Eq. 3 is first taken over the index k , calculations for d_{00k} are made starting with $k = 0$. Calculations then proceed with $k = 1$, $k = 2$, $k = 3$, $k = 4$, $k = 5, \dots$ and so on. When a distance, d_{00k} , is obtained that lies outside the boundary, the summation over k is terminated. What this process does is take the summation for all of the images found along one line in the circle.

When this process is repeated over the index j , the summation is made for

all of the images that lie inside the circular boundary. When the summation of all images lying inside of the circular boundary is repeated over the index i , the summation is made for all images that lie inside of a spherical boundary. When this is done the optimized summation is completed.

5 Performance and Evaluation of the Algorithm

Testing of this optimized algorithm was carried out and a comparison to the algorithm written by Allen and Berkley was made. The computer that was used to do this had a 2.2GHz AMD Athlon 64 3500+ CPU, 2x512MB of DDR 400 memory(3-3-3), an 800 MHz FSB, and Windows XP. Both algorithms were written in FORTRAN77 and both were compiled using the GNU G77 compiler.

Allen and Berkley's algorithm calculated distances for 10,648 images that comprised a rectangular lattice. In contrast, the optimized algorithm calculated distances for 3,086 images that comprised a spherical lattice. Both algorithms found a total of 2,725 images to be within the timeframe of interest, and both algorithms used these 2,725 images to construct the output impulse response.

To determine the execution time, each algorithm was programmed to run 80,000 times. When this was done, it was found that the program using Allen and Berkley's algorithm took 112 seconds to run, while the program using the optimized algorithm took only 13 seconds to run. This works out to a reduction of 88%.

To compare the output of the two algorithms, Allen and Berkley's code was modified so that the output data had the same normalized metric scale as the optimized algorithm. When this was done, it was found that the two algorithms output data that was nearly identical. Of the differences that did exist, they were both very small and consistent with floating-point round-off error.

6 Conclusion

Simple unsorted look-up tables reduce the number of calculations needed to compute both the total distances and the total pressure reflection coefficients. The use of sorted look-up tables reaps an added benefit by reducing the total number of distance checks. Usage of these two techniques together results in a substantially reduced computation time, and thus has potential applications for real-time processing as well as other computation intensive tasks.

References

- [1] Kuttruf H. Room Acoustics(4th Ed). Oxfordshire UK: Taylor and Francis; 2000

- [2] Gibbs BM, Jones DK. A simple image method for calculating the distribution of sound pressure within an enclosure. *Acustica* 1972; 26:24-32.
- [3] Allen JB, Berkley DA. Image method for efficiently simulating small-room acoustics. *J Acoust Soc Am* 1979; 65(4):943-950.
- [4] Borish J. Extension of the image method to arbitrary polyhedra. *J Acoust Soc Am* 1984; 75(6):827-1836.
- [5] Dalenback BL. Room acoustic prediction based on a unified treatment of diffuse and specular reflection. *J Acoust Soc Am* 1996; 100(2):899-909.
- [6] Kulowski A. Algorithmic representation of the ray tracing technique. *Applied Acoust* 1985; 18:449-469.
- [7] Wright JR. An exact model of acoustic radiation in enclosed spaces. *J Audio Eng Soc* 1995; 43(10):813-820.
- [8] Funkhouser T, Tsingos N, Carlbom I, Elko G, Sondhi M, West JE, Pingali G, Min P, Ngan A. A beam tracing method for interactive architectural acoustics. *J Acoust Soc Am*. 2003; 115(2):739-756.
- [9] Kompis M, Dillier N. Simulating transfer functions in a reverberant room including source directivity and head-shadow effects. *J Acoust Soc Am* 1993; 93(5):2779-2787.
- [10] Algazi VR, Duda RO, Duraiswami R, Gumerov NA, Tang Z. Approximating the head-related transfer function using simple geometric models of the head and torso. *J Acoust Soc Am* 2002; 112(5):2053-2064.
- [11] Ajdler T, Sbaiz L, Vetterlib M. Dynamic measurement of room impulse responses using a moving microphone. *J Acoust Soc Am* 2007; 122:16361645.
- [12] Kim T, Attias HT, Lee SY, Lee TW. Blind Source Separation Exploiting Higher-Order Frequency Dependencies. *IEEE Trans on Audio, Speech* 2007; 15(1):70-79.
- [13] Nakatani T, Kinoshita K, Miyoshi m. Harmonicity-Based Blind Dereverberation for Single-Channel Speech Signals. *IEEE Trans on Audio, Speech* 2007; 15(1):80-95.
- [14] Kristiansen UR, Krokstad A, Follestad T. Extending the Image Method to Higher-Order Reflections. *Applied Acoust* 1993; 38:195-206.
- [15] Foco M, Polotti P, Sarti A, Tubaro S. Sound Spatialization Based on Fast Beam Tracing in the Dual Space 2003; DAFx-03:Proc of the 6th Int Conference on Digital Audio Effects.
- [16] Schröder D, Lentz T. Real-Time Processing of Image Sources Using Binary Space Partitioning. *J Audio Eng Soc* 2006; 54(7/8):604-619.
- [17] Duraiswami R, Zotkin DN, Gumerov NA. Fast Evaluation of the Room Transfer Function Using Multipole Expansion. *IEEE Trans on Audio, Speech* 2007; 15(2):565-576.

A Appendix

```
//-----  
// C source file to calculate the  
// impulse response of a box-shaped room  
//-----  
  
#include <stdio.h>  
#include <math.h>  
#include <stdlib.h>  
  
//-----  
// Data structure to hold data for a particular dimension  
struct axisVar{  
    double w;          // room width  
    double r;          // distance from receiver to origin  
    double s;          // distance from sound source to origin  
    double Bmw;        // reflection coefficient for the wall  
                        // to the left of the origin  
    double Bw;         // reflection coefficient for the wall  
                        // to the right of the origin  
    int n;             // highest order reflection for the  
                        // look-up tables  
    double * distSq;   // pointer to square-of-the-distance  
                        // look-up table  
    double * refCoef; // pointer to reflection coefficient  
                        // look-up table  
};  
  
//-----  
// Function to create both a square-of-the-distance look-up  
// table and a reflection coefficient look-up table.  
void make_tables(struct axisVar a){  
    double Limg=1, Rimg=1, DirSnd, RimgD, LimgD;  
    int i;  
  
    // Set the first value in reflection coefficient table  
    a.refCoef[0] = 1;  
  
    // Set the first value in the square-of-the-distance look-up  
    // table  
    DirSnd = a.s - a.r;  
    a.distSq[0] = DirSnd*DirSnd;  
  
    // Set initial values for left and right image distance
```

```

// sequences
LimgD = DirSnd;
RimgD = DirSnd;

// Create tables
for(i=1; i<=a.n; i++){

    // Set image distances and reflection coefficients
    // for the ith order images on the left and right
    if(i%2 != 0){
        LimgD -= 2*(a.w+a.s);
        RimgD += 2*(a.w-a.s);
        Limg *= a.Bmw;
        Rimg *= a.Bw;
    }
    else{
        LimgD -= 2*(a.w-a.s);
        RimgD += 2*(a.w+a.s);
        Limg *= a.Bw;
        Rimg *= a.Bmw;
    }

    // determine pattern for ith order echo pair and
    // set values for the two tables
    if((i%2!=0 && a.r+a.s<0) || (i%2==0 && a.r-a.s<0)){
        a.distSq[2*i-1] = LimgD*LimgD;
        a.distSq[2*i] = RimgD*RimgD;
        a.refCoef[2*i-1] = Limg;
        a.refCoef[2*i] = Rimg;
    }
    else{
        a.distSq[2*i-1] = RimgD*RimgD;
        a.distSq[2*i] = LimgD*LimgD;
        a.refCoef[2*i-1] = Rimg;
        a.refCoef[2*i] = Limg;
    }
}
}

//-----
// main function
int main() {
    double c, time, *h, dist;
    int length, fs, i, j, k, samp;

```

```

// declare three data structures to hold information for
// each dimension
struct axisVar x, y, z;

// set the speed of sound, the time length of impulse
// response, and the sample rate of impulse response.
c = 343;
time = 1;
fs = 5000;

// find the length of impulse response in number of samples
length = (int)((float)fs*time);

// Set the room width, the source location, and the receiver
// location
x.w = 5; x.r = -2; x.s = 1;
y.w = 5; y.r = 4; y.s = 0;
z.w = 4.5; z.r = 4; z.s = -0.5;

// set two reflection coefficients for each dimension
x.Bmw = 0.9; x.Bw = 0.7;
y.Bmw = 0.9; y.Bw = 0.7;
z.Bmw = 0.9; z.Bw = 0.7;

// find the highest order reflection to be used in each
// look-up table
x.n = (int)ceil(c*time/(2*x.w));
y.n = (int)ceil(c*time/(2*y.w));
z.n = (int)ceil(c*time/(2*z.w));

// initialize square-of-the-distance look-up tables
x.distSq = malloc((x.n*2+1)*sizeof(double));
y.distSq = malloc((y.n*2+1)*sizeof(double));
z.distSq = malloc((z.n*2+1)*sizeof(double));

// initialize reflection coefficient tables
x.refCoef = malloc((x.n*2+1)*sizeof(double));
y.refCoef = malloc((y.n*2+1)*sizeof(double));
z.refCoef = malloc((z.n*2+1)*sizeof(double));

// create two tables for each dimension by calling the
// "make_tables" function
make_tables(x);
make_tables(y);

```

```

make_tables(z);

// initialize array for the impulse response
h = calloc(length, 8);

// use the look-up tables to calculate the impulse response
for(i=0; i<2*x.n+1; i++){
    for(j=0; j<2*y.n+1; j++){
        for(k=0; k<2*z.n+1; k++){

            // find the distance from the ijkth image to the
            // receiver
            dist = sqrt(x.distSq[i]+y.distSq[j]+z.distSq[k]);

            // find the sample in the impulse response that the
            // image with distance "dist" will occur at.
            samp = (int)(dist*(double)(fs)/c+0.5);

            // If the echo from the ijkth image occurs at a time
            // that is beyond the range of the bound of the impulse
            // response, then all of the images in one strip of the
            // circle have been tested, and the loop is terminated.
            // Note how a "break;" statement differs from a
            // "continue;" statement.
            if(samp >= length)
                break;

            // Add echo to impulse response array
            h[samp] += x.refCoef[i]*y.refCoef[j]*z.refCoef[k]/dist;
        }

        // If the loop over 'k' broke with k==0, then all of the
        // images in "the circle of interest" in the plane of i
        // have been tested and there is no need to find 'dist'
        // for any higher values of j in the current i plane.
        if(k == 0)
            break;
    }

    // If the loop over 'j' broke with j==0, then all the image
    // within "the sphere of interest" have been tested and there
    // is no need to find 'dist' for any higher values of i.
    if(j == 0)
        break;
}

```

```
}

// free allocated memory
free(h);
free(x.distSq);
free(y.distSq);
free(z.distSq);
free(x.refCoef);
free(y.refCoef);
free(z.refCoef);

return 0;
}
```