BEE08:02

Image Theory
Applied to Virtual Microphones

Carlos Hernández Matas
Diana Gómez Olmedilla

Department of
Signal Processing
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This degree project is submitted to the Department of Signal Processing, School of Engineering at Blekinge Institute of Technology. This thesis is 15 ECTS credits, equivalent to 10 weeks of full time studies.

Contact Information:
Author(s):
Carlos Hernández Matas
E-mail: carlos@hernandezmatas.com
Diana Gómez Olmedilla
E-mail: Diana.Gomez.Olmedilla@gmail.com

University advisor(s):
Dr. Mathias Winberg
Department of Signal Processing

Department of
Signal Processing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : +46 457 102 45

ABSTRACT

Virtual sensing is the art of measuring a quantity at a certain spatial position without having a physical sensor placed at that exact same position. In practical applications, this technique is sometimes very useful, when it is not feasible to put the sensor at the position where the physical quantity should be measured. For example, a virtual microphone could be considered in the following scenario: The sound level, preliminary school environment, should be studied at the different students head positions. In order to get statistical relevant data, the measurement has to be carried out over a period of time. However, it is not a feasible approach to use a large number of microphones hanging from the ceiling at the students head level, over a longer period of time for obvious reasons. One thoughtful solution, in such a situation is to put microphones at the walls and close to the ceiling and using the virtual technique to calculate the noise level at the students head positions based on the measure data. An advantage in such large setup is that it often requires less physical sensors than the number of virtual measurement positions and at the same time features a better validation and sanity check of the data. The most popular application areas for virtual microphones are in active noise control (ANC) or active structural acoustic control (ASAC), where the aim is to move the zone of quietness away from the physical error microphones to the desired location of maximum attenuation. However, there is an important difference between virtual sensing for active control and virtual sensing for a pure measurement situation concerning the delays introduced by virtual sensing technique.

This specific thesis work deals with the calculus of all the possible paths and reflections followed by a sound wave between the source and a given destination point, by using virtual transducers in an enclosed environment where reverberation exists. The aim of this work is to determine how to calculate that impulse response in a rectangular room in which the walls have uniform reflection coefficients and there isn't any kind of furniture inside. Besides, this project specifies a simple way to make this calculus taking into account several sources and destinations.

Keywords: acoustics, image theory, geometry, reverberation, algorithm.

A NUESTRAS FAMILIAS Y AMIGOS

Table of Contents

Introduction

Since old times, architectural acoustics has been used. Greek and Roman civilizations used to study the amphitheatre acoustics in order to improve the sound quality in plays, speech and music concerts by carefully planning the architectural design of the playground. The design involved the use of different reflective materials, such as different kinds of rocks and concrete and the construction of a ceiling over the actors area; besides, copper vessels were placed in specific areas over the stage so as to perform as resonators, distributing the sound to the upper seats [1]. All this aimed to keep a uniform sound pressure level in the whole playground.

At the end of the XIX century, modern room acoustics was born thanks to Wallace Clement Sabine, a Harvard physics teacher, who was the first to study the reverberation effects inside enclosure spaces and defined the reverberation time like we still know today. Although several physicists have tried to improve his studies and equations, such as Eyring and Millington, Sabine's principles remain intact nowadays.

From 60's, due mainly to the development of the computers, several acoustics engineers began to study computing applications to model the sound behavior in a room. This way physicists like Jont B. Alley and Berkley [10] published their algorithm to simulate the sound performance in small rooms based on the geometric image theory method. From then on, few acoustic engineers – like A. Krostad, M. Vorländer, P.A. Nelson, etc… have stated their approaches to model the enclosure spaces from the previous method.

Aside from the physical acoustics studies, in 1933 Harvey Fletcher and W. A. Munson published their conclusions about experiments on loudness carried out in Bell Telephone Labs [2]. Since then, it is also possible to get to know the psycho perception of the sound from the knowledge of the human hearing process. Although in the present project we will focus in the physical part, for some acoustics uses it is necessary to take a look at the human perception to save resources and improve the system quality.

From all the different physical methods to carry out the study of the sound behavior in a room, we have chosen the geometrical image theory method given its simplicity and intuitiveness. Firstly we will simulate the ray tracing approach and from there, we will model the same conditions with the image method to show its computational advantages.

The image method can be used for echoes treatment, enhance the sound, implement new loudspeakers distributions, buildings insulation and setting up, sound compression… [3]

We, as members of the Virtual Microphones Research Group at BTH, have developed this thesis as a tool for our colleagues to be able to predict the sound pressure in reverberant environments, so that they can plan the optimal set up for the virtual microphone technique.

Chapter 1: Fundamentals of Acoustics

The first decision that has to be made when working with sound particles is how they should be treated for a specific use: are we handling waves or rays? According to both categories, we can apply either wave or rays based methods.

On one hand, if we work with sound waves and want to model an enclosure space, the most known theories that can be applied are finite element method (FEM) and boundary element method (BEM). Both studies should be focus in finding the room resonance frequencies, pressure distribution within the modal planes, decreasing of the reverberant modes… These methods are only valid for very small rooms such as cages, for monitoring single frequencies. Therefore, given that the human hearing perception works in octave bandwidths and the size of the common rooms and auditoriums is always bigger than 15 m3, this approach is not the suitable one in our situation.

On the other hand, handling sound rays we can apply the geometric method, which is more flexible with room dimensions and frequency ranges, as well as being more intuitive. This method is more suitable for our study as it supplies increasing accuracy and advantages as the wavelength gets bigger compared to the room size. F. Alton Everest deeply explains this matter in The Master Handbook of Acoustics [4].

1.1 Sound behaviour

According to what we explained in the above paragraph, we will now distinguish between the wave performance in an ideal environment and in an enclosure space.

1.1.1 Ideal wave propagation in open field

In an ideal situation, the wave propagates on free field, i.e. Earth's surface. The sound spreads with homogeneous intensity from the source to all directions, resulting in a spherical wavefront. One of the main consequences of this situation is the property of the spherical divergence, which means that every time the wave doubles the distance covered, that sound intensity decreases inversely proportional to the distance, so we have that:

As $I \sim p^2$ therefore, when the distance is doubled the pressure decays by r^2, which in decibels is equivalent to a decay of 6 dB.

Where:
I is the wave sound intensity measured in W/m^2.
p is the sound pressure in Pa.
r is the distance covered by the sound in meters.
This property is called the spherical divergence. In this ideal situation, that spherical divergence should be applied to a sphere, so the ratio is the sphere radio $4\pi r$.

But the wave behaviour doesn't remain loyal to this property through all its way: when the sound wave reaches a certain distance, the energy due to the source power begins to lose importance against the energy due to reflections on the Earth's surface. At this point the wave doesn't propagate in a free field any more: it travels through a reverberant field, where the

source power and the surface absorption have the same impact on the total sound pressure level, no spherical divergence is considered and a study at every single frequency should be developed to achieve accurate results. This decisive distance is called the critical distance, Dc, and at this point the contribution of both the free and the reverberant field is the same. Hopkins-Stryker developed a set of equations that fully explain these sound phenomena (1), as follows:

$$SPL_t = L_w + 10\log\left(\frac{(Q(\theta,\phi))}{(4\cdot\pi\cdot r^2)} + \frac{4}{R}\right) \quad (1) \rightarrow SPL_t = SPL_d + SPLr$$

Where:
L_w is the power level of the source [dB]

$\frac{(Q(\theta,\phi))}{(4\cdot\pi\cdot r^2)}$ This part of the equation corresponds to the contribution to the direct sound (free field), which depends on the frequency and directivity of the source $Q(\theta,\phi)$

$\frac{4}{R}$ This is the contribution of the reverberant level, and depends on the walls absorption according to the wavelength and room volume. R is the acoustic room constant, and it is based on the previous parameters and measured in m^2 in the IS. $\overline{\alpha}$ is the average sound absorption coefficient in the room.

$$R = \frac{S\cdot\overline{\alpha}}{(1-\overline{\alpha})}$$

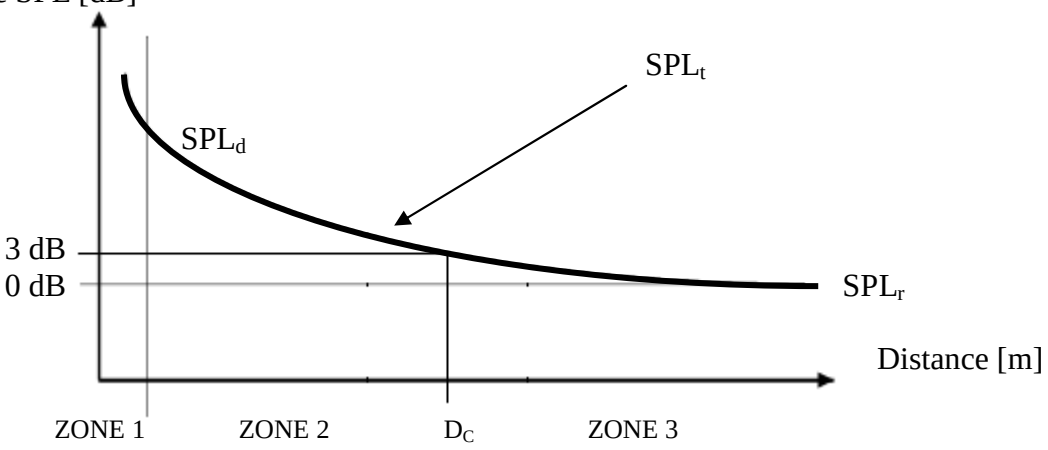


Figure 1. Sound pressure variation according to critical distance.

ZONE 1: Loudspeaker close field. The measurements at this distance are not useful given that at this distance the sound particles are chaotically distributed.
ZONE 2: Direct sound field. The contribution of the direct signal (SPL_d) is much bigger than the contribution of the reverberant level (SPL_r). Here the spherical divergence is valid.
D_c : At this distance, both the direct and reverberant levels have the same value therefore, the total level is SPL_t + 3dB.
ZONE 3: Reverberant field. In this area, the SPL_t is basically due to the reflections on the walls. As explained in (1), the total level depends directly on the acoustical properties of the room – R. If the reverberant field is fairly stable, we can say that it is a diffuse sound field.

In the previous figure, the theoretical sound areas are shown, in each case study the observation must focus in the suitable area; as it is in our case - enclosure space with reverberation - we will state that the most proper study zone is ZONE 3 and we will consequently calculate the impulse response for that area.

NOTE: In Figure 1, the y-axis is plotted in relative levels, just to show the general behavior of the sound and the weight of each area in the total sound level.

According to (1) the critical distance is obtained when the contribution to the direct and reverberant sound level is equal.

$$for\ r = D_c \rightarrow \frac{(Q(\theta,\phi))}{(4\cdot\pi\cdot r^2)} = \frac{4}{R} \rightarrow D_c = \sqrt{(\frac{(R\cdot Q(\theta,\phi))}{(16\cdot\pi)})} \quad (2)$$

It is extremely important to know these parameters in our case study because this knowledge can show us the relation between reverberant and direct sound; in 3.2 Section, R will be carefully explained according to Sabine's equation.

1.1.2 Enclosure spaces

In our case study, the ideal conditions shouldn't be applied in such way: first of all, the spherical divergence can only be applied in free field, which means that if we have any reflections the sound pressure is not really inversely proportional to distance^2, so the relation of the 6 dB is not accurate any longer. In a hypothetical case, close to the source, there could be a spherical wavefront because the contribution of the sound power to the total level is still bigger than the reverberant one, unfortunately this assumption is not practical in real acoustical measurement. Therefore, the analysis of the sound total pressure in a room shouldn't be based in spherical wavefront.

Secondly, there are two ways to study the sound distribution in a room: either making such study using the wave based methods or with the ray based methods. As it has been explained in the beginning of the chapter, in this project the sound particles are treated as a ray so that we can use the geometrical methodology. The advantages of this choice are that the study can be applied to grater ranges of frequencies and there is more flexibility with the size of the room we are studying.

As it has been properly stated in [5], the geometrical method is used to simplify the sound behavior for high frequencies, given that for those wavelengths (limited in comparison with the room size) the sound rays behave exactly like the light rays [6]. So that, the incident angle is the same as the reflected angle, and it can be considered that the reflections are specular. Which means that no incidence angle or interference process has to be taken into account and the diffraction is irrelevant.

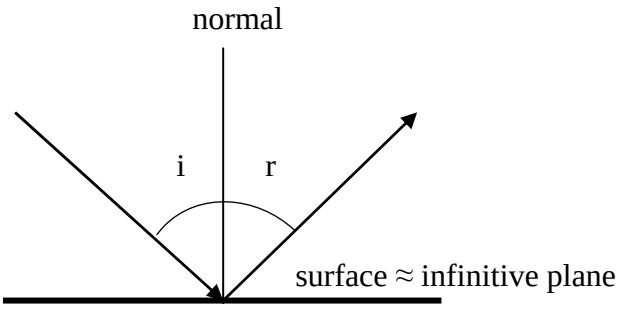


Figure 2. Law of reflection.

Applying the previous law, we will develop our approach basing the total sound energy in the reflection coefficient, β, which is directly related to the absorption coefficient as explained next (5). Consequently, our analysis can be done by suitably summing flat rays.

1.2 Building Materials

Given that the purpose of this project is the analysis of the sound in a room, the main absorption factors responsible for the energy decay in this situation should be defined:

1. Building materials, furniture and audience inside the room.
   For this project, the energy decay in the stay depends directly on the walls materials, that is, we will focus our approach in the absorption coefficients of those materials. In case there is furniture or people inside the room, the specific absorption area should be studied according to Sabine's Law. In our case, an empty room will be studied, or approximated as empty.
2. Dispelling due to wave propagation at high frequencies.

$$\mu = \frac{85}{\theta} \cdot t^2 \cdot 10^{(-4)} \qquad (3)$$

μ dispelling coefficient
t temperature ≈ 20 ºC
θ air humidity between 30 % and 80 %

   This factor is outstanding in huge environments such as stadium and open air auditorium. Given that this project focus in enclosure spaces with smaller dimensions, μ will be considered constant.
3. Dispelling due to wave spherical divergence. Already argued in 1.1.1.

The absorption coefficient (α) represents the relation between the absorbed energy and the incident energy that hits a wall (4). Its values are from 0 to 1 –which is the alpha value of an anechoic chamber-, representing 1 a perfect absorbent material. This way, depending on alpha's value, a proper equation should be applied in order to get the reverberation time, as it will shown in 3.2 Section.

Besides, the approach can also be based on the reflection coefficient (β), as follows:

$$\alpha = \frac{(Absorb\ Energy)}{(Incident\ Energy)} \quad (4) \qquad \alpha = 1 - \beta^2 \quad \rightarrow \quad \beta = \pm\sqrt{(1-\alpha)} \quad (5)$$

Usually, when the study of the absorption materials performance is carried out, a precise observation on the central octave bandwidths frequencies should be done, that is measuring at 125, 250, 500, 1000, 2000 and 4000 Hz. However, this issue won't be examined in this project, given that we will concentrate on frequencies from 4kHz onwards, where the absorption coefficient remains constant.

The materials are usually included in three main types of materials: porous materials, Helmhotlz type sound resonators and plates – simple sound resonators. Each material should be used depending of the situation, but keeping in mind that the more absorption is achieved for bigger frequencies and that when manufacturing a set of materials, the thicker, the more absorbent.

Although the alpha values are already tabulated for the most common building materials (see Appendix I to check the most used), it is possible to carry out a calculation to get to know that coefficient i.e. new materials or set of different types of materials. This process is very easy provided that the suitable equipment and facilities are used. Introducing the material in a reverberant room and make the following measurements are enough:

$$\alpha_m = \frac{0.16 \cdot V}{S_m} \cdot \left(\frac{1}{T_1} - \frac{1}{T_0}\right) \quad (6)$$

α_m is the unknown material's absorption coefficient .
S_m is the material's surface [m^2].
T_1 is the reverberant time (RT_{60}) measured with the tested material inside the reverberant room.
T_0 is the reverberant room's RT_{60}.

This is the usual approach, although new methodologies are being reviewed [7].

Chapter 2: Problem definition & Methodology

The main purpose of this project is to carry out the study of the sound paths in an enclosure space, in an intuitive and straightforward way suitable for common audience. Within the approach both calculations and room representations will be shown for a better understanding.

The process that will be held is the observation of the energy level at a given point, under one source performance. For that calculus, we'll use the image theory methodology, but to develop that algorithm firstly, we'll recreate the ray tracing algorithm, due to its similarities and the easiness of the latter.

- Equipment requirements

For this research, the transducer set is based on a loudspeaker, radiating power of 1W, and a microphone. This equipment must have an omnidirectional performance, radiating and receiving the energy equally in all directions.

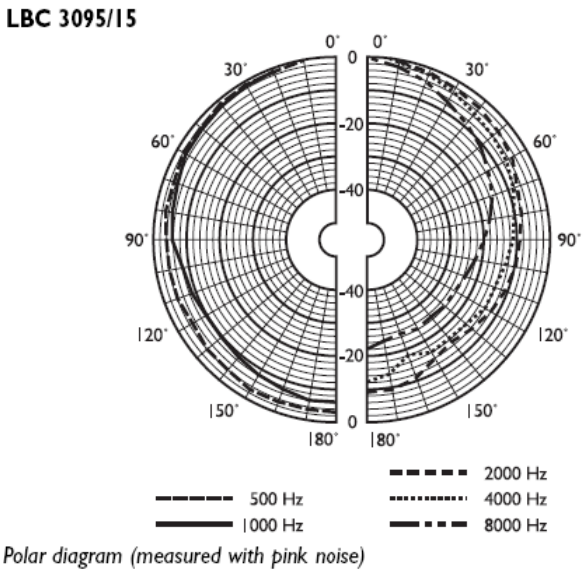


Figure 3. Polar diagram of a hypothetical omnidirectional microphone [8].

- Room statements

The room observed is rectangular with parallel, flat and rigid walls. No windows, doors, furniture, or audience will be taken into consideration along this approach so that a uniform absorption will be estimated for each surface. The room dimensions are not easily mathematical related so as not to excite nodal modes, the most common proportions between ceiling, walls and floor used in this kind of studies are for example, Boner proposal 1:1.26:1.59,Volkmans ratio 2 : 3 : 5, Sepmeyers proportions, … [4]

Given that we are examining an office environment, the volume of the space will be bigger than 10 m^3 and smaller than 5000m^3.

Chapter 3: Theoretical work

Once all the statements are made, a meticulous observation of the sound distribution in the room will be carried out by using both the ray tracing technique and the image theory.

3.1 Ray tracing

Counting on that the source emits the sound with the same probability in all directions – omnidirectional source - , the ray tracing method makes possible to carry out an accurate study of each ray path, from the source to the observation point listing all the walls it crashes. This way, it is very easy to follow the sound ray history and calculate the remaining energy after all the impacts, according to the walls absorption coefficient for each frequency [9]. As an example, in the following picture the direct ray path (purple) and two first order reflections (red) paths are shown:

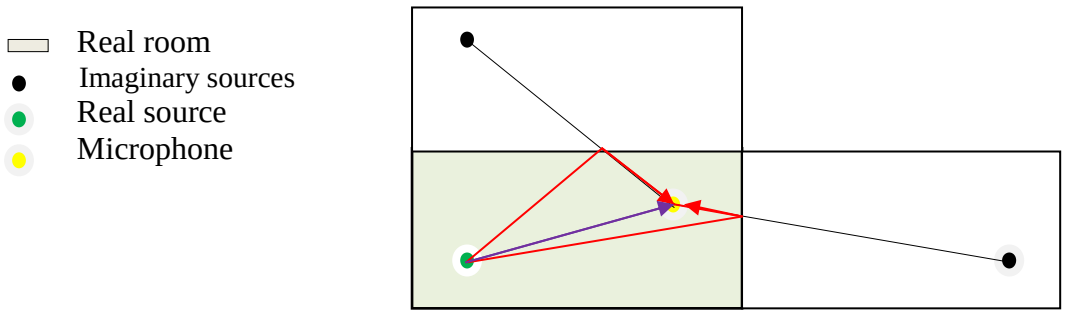


Figure 4. Representation of the sound ray paths' according to the first order reflections on the ceiling and right wall.

As explained before in 1.1.2 Section, in our case study only specular reflections will be considered –Figure 2. Furthermore, as this approach is made for high frequencies, it is considered that the walls absorption coefficients are constant and independent on the frequency range. Although the last statement seems to be at first sight a tough approximation, for high frequencies –over 4kHhz- it works pretty precisely.

The advantages of this method are that it is very intuitive and straightforward to implement, given that only crashes and energy decay are computed. Nevertheless, as with this technique each imaginary source has the same significance in the total energy, the computational time increases exponentially as more reflections are added. Besides, another drawback is that this technique is only valid for static observation points.

At the beginning we studied the wave as if its propagation was spherical (7), but after having done deeper research we came to the conclusion that inside the room the wave cannot propagate with an spherical wavefront because the energy level due to reverberation has more importance than the level that comes directly from the source [4].

$$E_k = W \frac{\left((1-\alpha_1)^i \cdot (1-\alpha_2)^j \ldots (1-\alpha_6)\right)^m}{\left(4\pi c\, l_k^2\right)} \qquad (7)$$

Where, the distance to the microphone

$$l_k = \sqrt{((x_k - x_0)^2 + (y_k - y_0)^2 + (z_k - z_0)^2)} \quad (8)$$

(x_k, y_k, z_k) is the position of the imaginary source, k order.
(x_0, y_0, z_0) microphone position.

α_b is the absorption coefficient of each of the 6 walls, with b = 1, 2, 3, 4, 5, 6 and (i, j, …m) their respective number of impacts.

In conclusion, in the final approach the equation (7) is not used. Instead, the following formulae (9) is applied as well as (8):

$$E_k = W \frac{(\beta_1^i \cdot \beta_2^j \dots \beta_6^m)}{(l_k)} \quad (9)$$

β_b is the reflection coefficient of each of the 6 walls, with b = 1, 2, 3, 4, 5, 6 and (i, j, …m) their respective number of impacts.

The minimum quantity of rays, N, that need to be calculated with this method are:

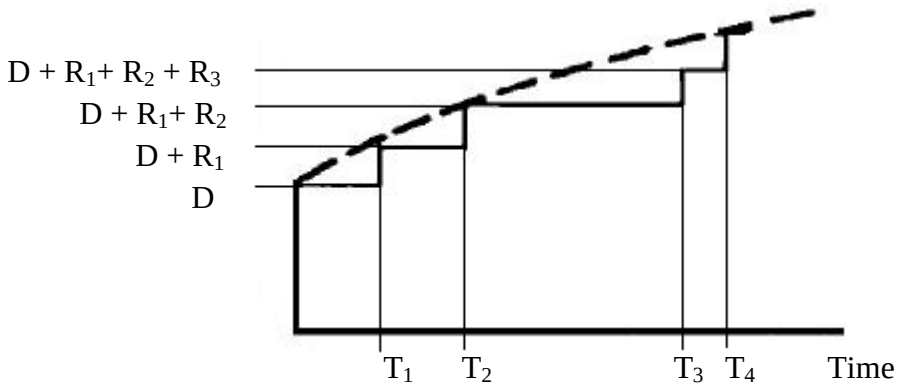
$$N \geq \frac{8 \pi c^2}{A} t^2 \quad (10)$$

c is the sound speed with value 340.29 m/s^2
t is the minimum computational time that should be chosen for a suitable measurement.
A is the room area m^2.

3.2 Image theory

Once the ray tracing program is implemented, if the sound distribution in the room is tested for different reflection orders, it is noticeable that when reached a certain number of orders –depending on the volume of the room and α of the walls- the contribution to the total energy doesn't vary very much. This observation yields that no infinity reflection orders are necessary to get a fairly good idea about the energy in the room. Furthermore, in most situations just up to 6^{th} order is needed. This issue apart from that in the real environments the microphone cannot see all the imaginary sources -due to the room geometry-, leads to the study of which reflections actively participate in the total energy

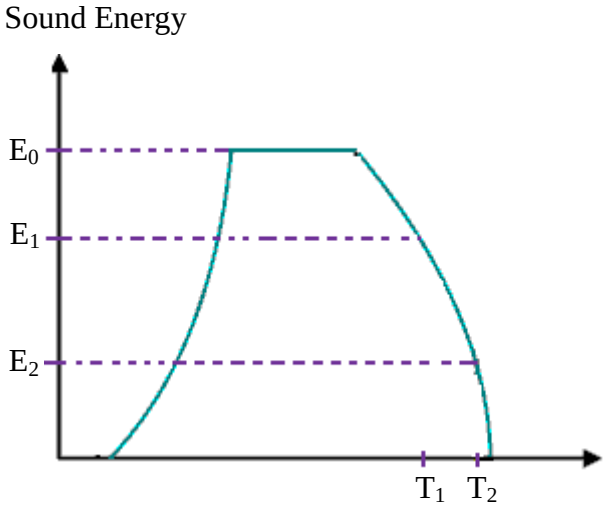
Where:
D is the direct ray
R_n are first order reflections



Figures 5. Energy evolution by adding new reflections [4].

The image theory states that the noteworthy imaginary sources are distributed inside a sphere with radio $c \cdot t$, where c is the sound speed and t is the time that should be properly calculated in order to take into account all the significant reflections. This time can be estimated either compromising accuracy and computational requirements –calculation speed and computer restrictions– or according to the reverberation time [10].

The reverberation time (RT_{60}) is the time that, once the source is stopped, takes the signal to reduce its energy in a million times - which means a sound level decreasing of 60 dB. Considering that the RT_{60} is calculated from the walls absorption coefficients and those have different behaviour according to the sound ray wavelength, the RT_{60} is defined at a specific frequency, changing its value depending on the frequency measured [11]. The physical explanation is:



$$E_2 = E_1 \cdot 10^{-6}$$

$$E = E_0 \cdot e^{\left(\frac{-13.8 \cdot t}{T}\right)}$$

$$L_2 = L_1 - 60\,dB \quad (11)$$

Figure 6. Reverberation time

For E_0 is the maximum sound level before ears pain; $T_2 = RT_{60}$

The study of the RT_{60} entails the application of the statistics theory as well as the geometrical methods. This is possible because we assume that the sound field in the room is diffuse given that the energy contribution due to reflections is bigger than the direct sound.

However, the statistic theory cannot be applied freely, the following requirements must be fulfilled in our study case:

- Omnidirectional sources and microphones.
- Waves directions with the same probability.
- Random waves phase distribution at each point.
- Energy in a point is the sum of the energy of all the rays that arrive at that point.
- Total energy is the whole energy in the room.
- Reflections studied statistically.
- Energy density evaluation measured in time domain.

The formula most widely used to empirically calculate the RT_{60}, is the one stated by Sabine in 1968. Nevertheless, recent studies have achieved the conclusion that Sabine's equation works pretty well for $\overline{\alpha}$ lower than 0.2, whereas the formula put forward by Eyring suits best for $\overline{\alpha}$ values from 0.2 to 1. Both equations are explained below:

$$A_t = \sum S_k \cdot \alpha_k$$

$$T_{Sabine} = \frac{0.16 \cdot V}{A_t} \rightarrow \alpha < 0.2$$

$$T_{Eyring} = \frac{0.16 \cdot V}{(-S \cdot \ln(1-\overline{\alpha}))} \rightarrow \alpha > 0.2 \quad (12)$$

Where:
A_t is the equivalent absorption area measured in m^2
V and S are the room volume and surface.
S_k is the wall surface, α_k the corresponding absorption coefficient.

The control of the reverberant time in an enclosed space is of vital importance, since it summarizes the room's sound performance as well as, how a hypothetical audience can listen in it. A suitable value of the RT_{60} can reinforce the musical presence, instruments glittering, and speech; whereas, too big values of RT_{60} can lead to even echoes effects. That's why a detailed acoustical study, mainly during the architectural design step, can avoid many hearing perception problems i.e. in an auditorium. The optimal RT_{60} values depending of the room use and frequency range have been tabularized for long, next, an example with the plots that the NASA handle while carrying out acoustical experiments:

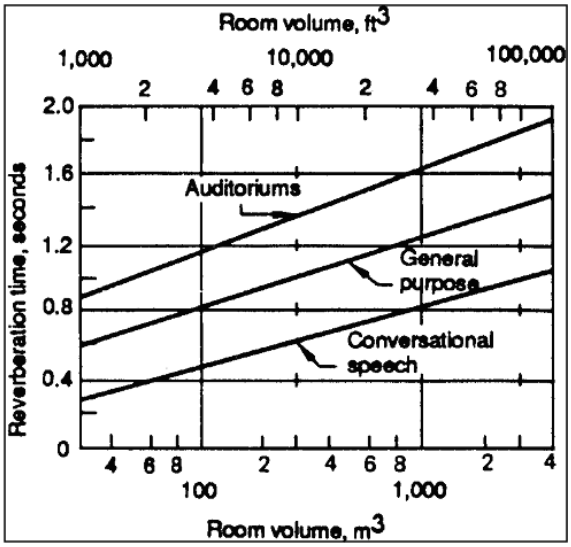


Figure 7. Optimal RT_{60} according to NASA [12].

Another outstanding parameter which defines the room acoustical properties is R:

$$R = \frac{\overline{\alpha} \cdot S}{(1 - \overline{\alpha})} \quad (13)$$

Using the Master Handbook of Acoustics as reference, we discovered that the image theory is not valid for all the range of human-hearing frequencies. It properly represents the sound from f_0 to 20 kHz, being f_0 dependent of the characteristics of the room and calculated by:

$$f_0 = 45000 \cdot \sqrt{\left(\frac{RT_{60}}{V}\right)} \quad (14)$$

As an example, in a room with dimensions 3 x 4 x 2.6 m. and absorption coefficient of 0.2 in each wall, f_0 = 5179 Hz, so the image theory method in this case is valid from 5.179 - 20 kHz. On the other hand, although the absorption coefficient of the surfaces depends on the frequency, it remains constant from medium-high frequencies –4kHz. With this data we can extrapolate that the impulse response obtained should be fairly correct for the frequency range between f_0 and 20 kHz.

In this case, with the image method, the minimum number of imaginary sources that must be calculated is:

$$N_{reference} = 4 \cdot pi \frac{(c \cdot t_{max})^3}{3\mathrm{V}} \quad (15)$$

That compared to (10), is obviously much smaller which allows saving a lot of computational time.

3.3 Impulse response

Getting to know the impulse response of a room allows the engineers to understand how the sound is behaving in that place. This knowledge is a powerful tool to avoid sound disturbances and correct building deficiencies.

The impulse response is usually observed through an echogram, which represents the sound arrival in a point, distinguishing between the direct sound and both early reflections -arrive up to 30 ms later than the direct sound- and late reflections –up to 50 ms later than the direct sound-, as it was documented by Cremer [13]. In this project, two impulse representations have been applied:

- Doak & Bolt representation is used to check the echo in an enclosure space. This is highly important in order to provide the audience a comfortable auditory. If the late reflections –which arrive 50 ms later than the direct sound- have higher level than the previous ones, it is considered to be echo which means a disturbing sound phenomena that should be avoid in most cases. The easier way to check if an echo is propagating in the room is by looking at the level of the late reflections: if a ray arrives to the microphone with considerably higher level than the previous ones, there must be an strange sound phenomena that should be thoroughly observed –such as in the example in Section 4.3.

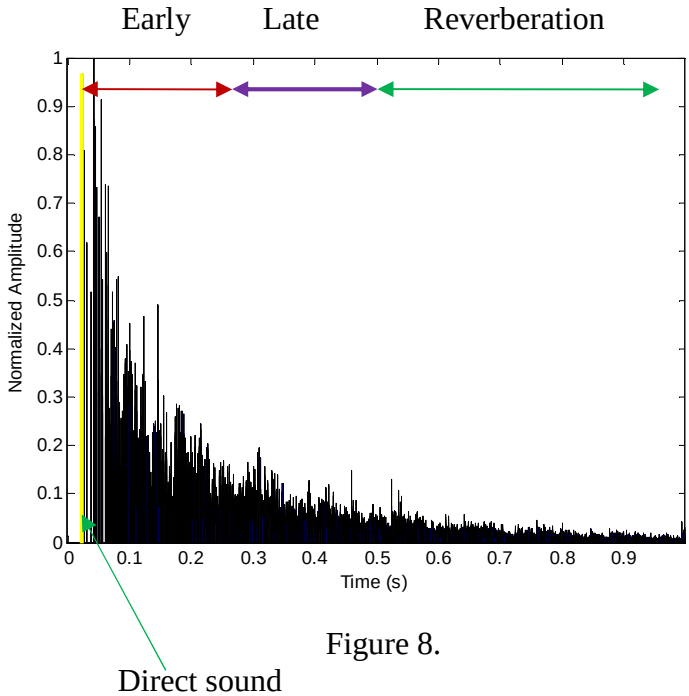


Figure 8.

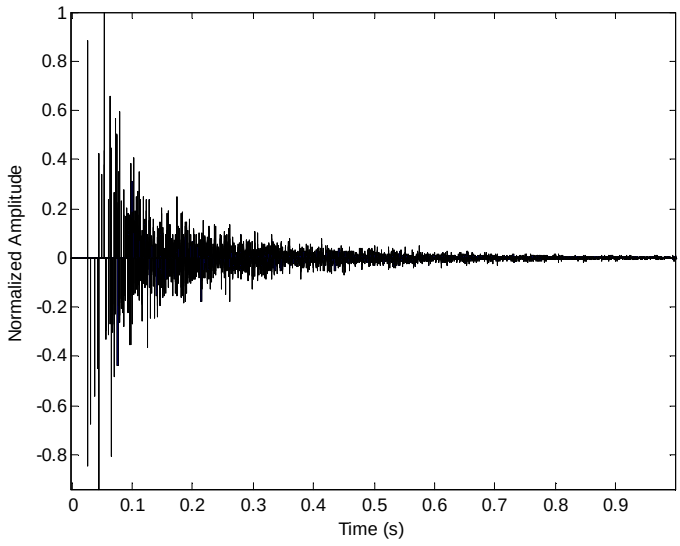


Figure 9.

Figure 8. Doak & Bolt impulse response representation.
Figure 9. Symmetrical impulse response representation.

In order to get the impulse response of the signal, we use the reflection coefficient (5) considering that the 50% of the reflections are positive and the other half are negative. We came to this conclusion after having made some test to our program with different percentages.

These representations give a clue about the material distribution inside the room, i.e. in Figure 10, the absorption coefficients for each wall are completely different so that the echogram shows an acoustical problem with even echo effect. In figure 11, the materials chosen are matched so as to get uniform reflections pattern, obtaining a logical sound response.

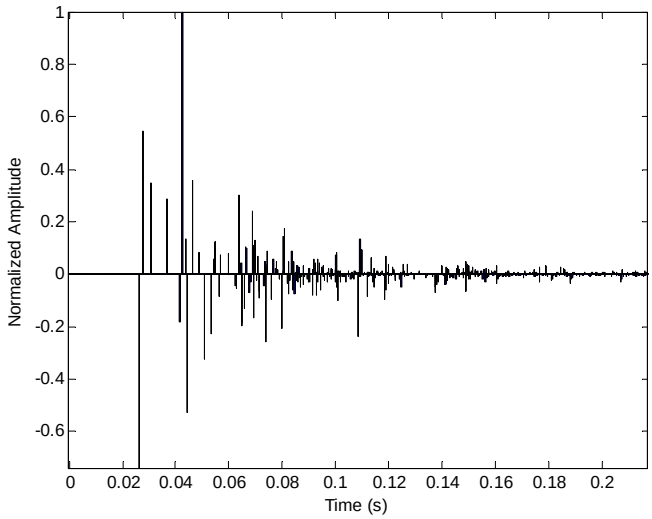


Figure 10.

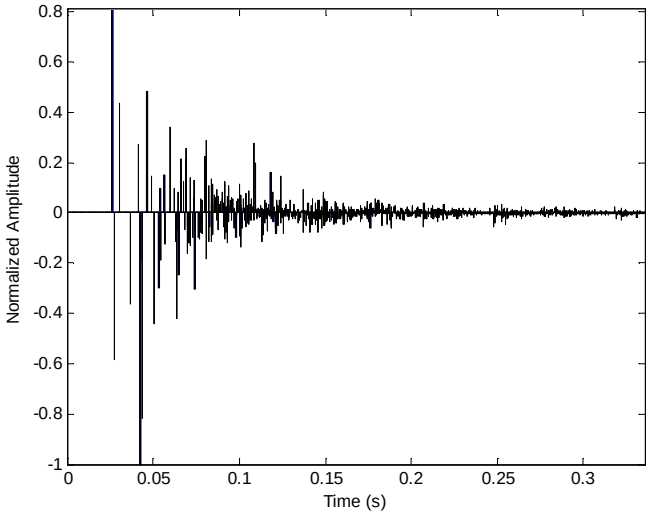


Figure 11.

Absorption coefficient (α)	**Ceiling**	**Floor**	**Wall 1**	**Wall 2**
Figure 10.	0.4	0.5	0.8	0.7
Figure 11.	0.3	0.6	0.6	0.3

Table 1. Impulse response comparison according to the walls materials selected.

NOTE: All the impulse response figures both in this Section and in Section 4.3 are seizes of the program imgthry3d.m that we have implemented for this thesis.

Chapter 4: Matlab implementation

Despite using Matlab for this project, the code was not optimized to take advantage of its special functions for managing arrays and matrixes, but having in mind the chance of porting easily this program to other languages, such as C.

4.1 Ray tracing

4.1.1 Ray tracing 2D

While we were beginning to develop the code, we decided to do our first approach from a geometrical way, which means, creating first the grid containing the position of the virtual sources along the space. Also, as the image theory method is built upon ray tracing, we started by doing a bidimensional representation of the ray tracing.

We began creating first the 2 axis -that is, x and y-, and then the arcs than goes from one axis to the other, always calculating just the virtual sources that were desired because of the amount of reflections -having the diagram a diamond shape-, not all the grid -which would have had a square shape.

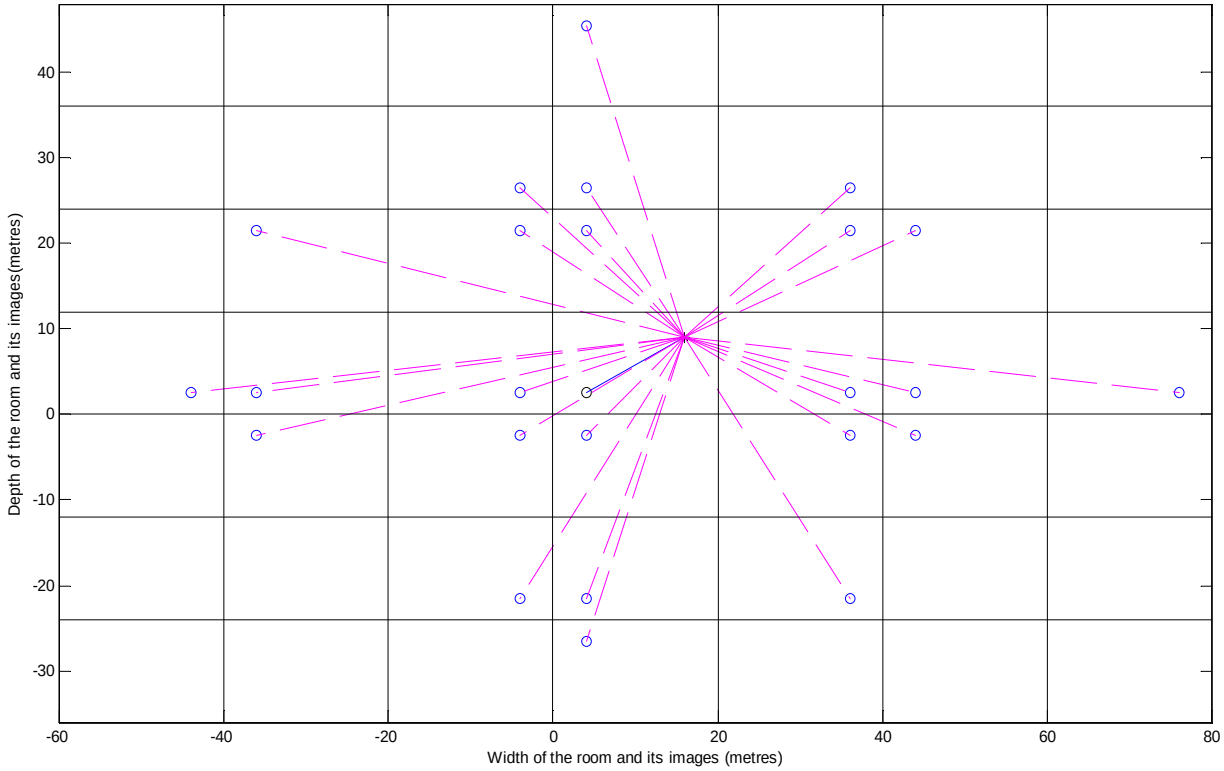


Figure 12. Ray tracing 2D with the following set up:

Room size [m]	**Source position [m]**	**Microphone position [m]**	$\overline{\alpha}$
[20 12]	[4 2.5]	[16 9]	0.3

Table 2. Simulation set up for Figure 12.

NOTE: The room dimensions have been chosen according to Chapter 2. Room Statements.

4.1.2 Ray tracing 3D

Once this was successfully fulfilled, the next step was to go on from the bidimensional image theory representation to the tridimensional one. For that, the two first steps were the same than for the 2-dimensional representation: that is, first we created the 3 axis and the arcs that joint them. The new problem here was to fill the pyramids existing between the different arcs, calculating, once again, only the desired virtual sources.

Then we created an algorithm to calculate the energy of the signal in the position of the microphone, and did a brief study on how the size of the room, the distance between the microphone and the real source, and the absorption coefficient affected the energy received.

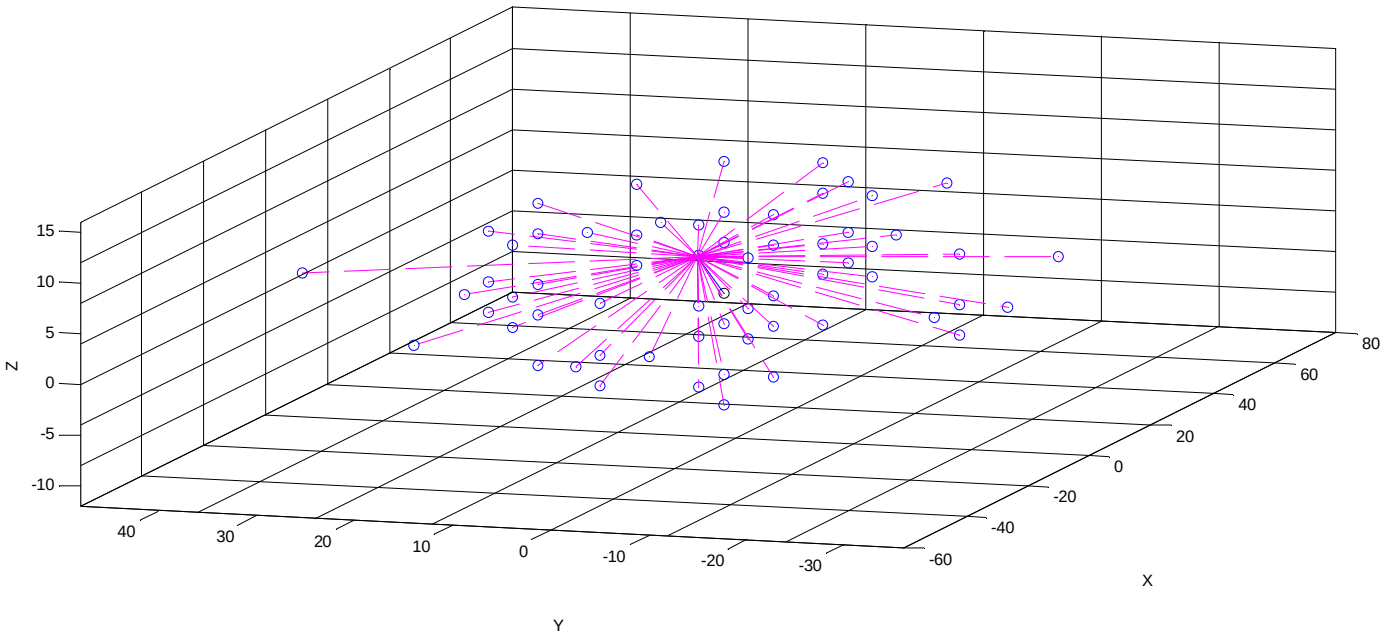


Figure 13. Ray tracing 3D with the previous set up

Room size [m]	**Source position [m]**	**Microphone position [m]**	$\overline{\alpha}$	**Total Energy [J]**	**Total dBSPL**
[20 12 4]	[4 2.5 1.5]	[16 9 3]	0.3	0.0014	37.0069

Table 3. Simulation set up for Figure 13.

Although in the representation a uniform absorption coefficient has been chosen for the room, it is possible to specify different absorption coefficient for walls, ceiling and roof.

As shown in Table 3., our implementation calculates the total sound energy at the observed point, by applying the following formula:

$$E_t = W \cdot \sum_k \left(\frac{\sum_i (rc_i)^{ci}}{d_k} \right) \quad (16)$$

Where:

W is the source sound power [W]
d_k is the distance attenuation for k, imaginary source, with values k = 1, 2, … N
rc_i is the reflection coefficient for the wall i. Given i = 1, 2, … 6

$\sum_{i=1}^{N} (rc_i)^{ci}$ is the wall absorption coefficient for the specific i.

4.2 Image theory 3D

After this, we then had to modify our algorithm to step into the image theory method. As here, we just needed the image sources included in a period of time, this, geometrically represented, takes into account the virtual sources included in a given sphere with the microphone as its center.

In a first approximation, we decided to calculate first all the virtual sources in the tridimensional grid, and then choose only the ones inside the sphere. As this method took a lot of processing time, we then decided to make a tridimensional diamond, precisely oversized to contain the sphere in it, and then take into account only the virtual sources inside the sphere.

In spite of the second method was fairly complicated –it involved calculating a sphere inside a diamond inside a cube-, compared to the first one -calculating a sphere inside a cube-, it proved to be from 3 to 4 times faster, so that, the one we finally chose to use.

Having the algorithm to calculate the virtual sources implemented, we did another brief study to take into account how the size of the room, the absorption coefficients and the radius of the sphere affected to the energy received in the microphone position. Despite this study, we set up the algorithm to use the reverberation time for the radius of the sphere. Also, we let the chance to the user to introduce its desired radius to make the calculations.

Here is the help of the function imgthry3d.m

```
%   imgthry3d: Calculates the impulse response of a room using image theory
%
%   function [h, time, tr, f0] = imgthry3d (room, source, mic, fs, plotIR, plotVS)
%   example of use
                [h, time, tr, f0] = imgthry3d ([3 4 2.6], [1 1 1], [2 4 1.6], [0.2 0.2 0.2 0.2], 10240, 0, 1, 0);
%
%   h is an array with the impulse response of the room for the given mic source and dimensions of the
%                room
%   time is the temporal axis of the impulse response; tr is the reverberation time of the room
%   f0 is the starting frequency from which this method is valid.
```

```
%   This method is valid in the following frequency range: f0 - 20 kHz
%
%   Room is an array with the coordinates of the room to be used, uses the following format
%               [roomX roomY roomZ]
%   Source is an array with the coordinates of the position of the source to be used, uses the following
%               format: [sourceX sourceY sourceZ]
%   Mic is an array with the coordinates of the position of the microphones  to be used, uses the following
%               format:  [micX micY micZ]
%    Absorption is an array with the absorption coefficients, uses the following format:
%               [x-z-walls y-z-walls ceiling floor]
%    fs is the sampling frequency
%    timec is the lenght in seconds for the impulse response calculations.
%    If  '0', the program uses the reverberation time, unless the reverberation
%               time is longer than 1 second. In that case, it will make the calculation for 1 second.
%    plotIR indicates if you want to plot the impulse response
%    plotVS indicates if you want to plot the 3D representation of the virtual sources
%
%    Version 0.4.2
%    2007-2008, Carlos Hernandez Matas and Diana Gomez Olmedilla
%    Contact: carlos@hernandezmatas.com and Diana.Gomez.Olmedilla@gmail.com
%
```

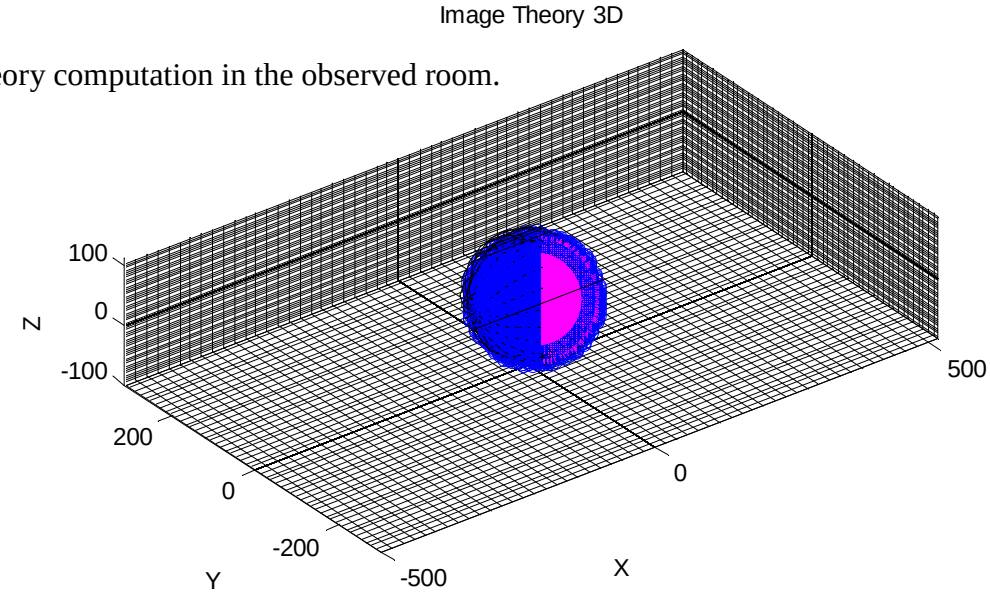


Figure 14. Image theory computation in the observed room.

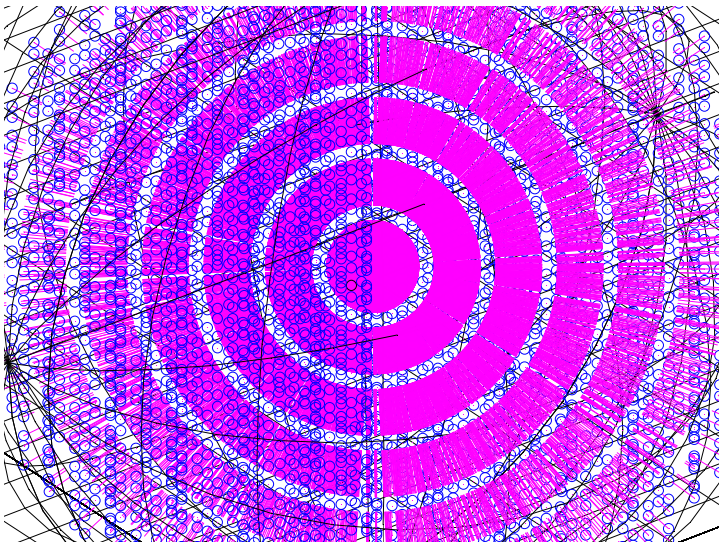


Figure 15. Zoomed Image Theory.

4.3 Impulse response

Now, having completely finished with the geometrical part, the next step was to calculate the impulse response function. Knowing the time delay of the signal to arrive from the virtual source to the microphone, and also the energy it carried, it was only a matter of adding them timely separated:

$$IR = W \cdot \sum_{k} \left(\frac{\sum_{i} (rc_i)^{ci}}{d_k} \cdot \delta\left(n - \frac{d_k}{c}\right) \right) \tag{17}$$

The previous equation is obviously based on (16), with:

$\delta\left(n - \frac{d_k}{c}\right)$ This term gives the position of the ray energy. Based on the attenuation distance for each k, imaginary source.

For the representation of the impulse response, we considered half of the reflections positive and the other 50% as negative. We chose those parameters based on a statistical study that we performed in which we simply tried many combinations of percentages between positive-negative ray impacts, coming to the conclusion that 50% each has a fairly good response.

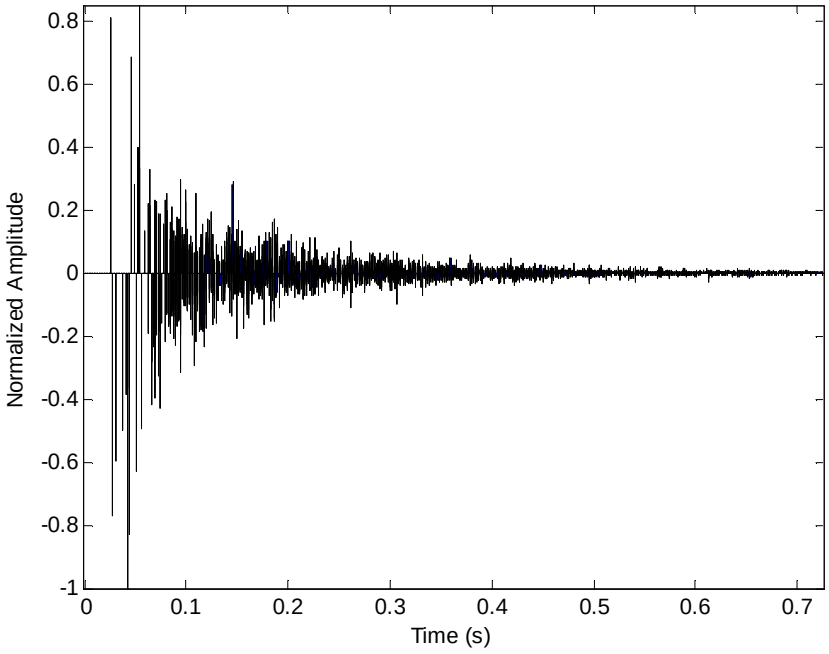


Figure 16. Symmetrical impulse response for the following set up:

Absorption coefficient (α)	**Ceiling**	**Floor**	**Wall 1**	**Wall 2**
Figure 16.	0.25	0.25	0.25	0.25

Table 4. Materials distribution used for simulation in Figure 16.

4.4 Multi Image theory implementation

Finally, we implemented another program based on the previous one in such a way that given any number of real sources and microphones, it calculates the impulse response between every source with every microphone. As this program performs as many combinations source-microphone as it has been introduced through the command line, a capture is not shown above because the result is a set of image theory observations for every couple source-microphone, even though the results are very useful in practical situations.

Here is the help of the function multi_imgthry3d.m

```
%   multi_imgthry3d: Applies imgthry3d to several microphones (mics) and sources
%
%   function [h, time, tr, f0] = multi_imgthry3d (room, sources, mics, absorption, fs, plotIR, plotVS)
%   example of use:
[h, time, tr, f0] = multi_imgthry3d ([3 4 2.6], [1 1 1; 1 1 2], [2 2.5 2; 2 4 1.6], [0.2 0.2 0.2 0.2], 44100, 0, 1, 0);
%
%    h is a matrix with the impulse responses of every combination of sources and mics following this order:
%                source1mic1
%                source1mic2
%                ...
%                 source1micN
%                ...
%                sourceMmicN
%    time is the temporal axis of the impulse responses; tr is the reverberation time of the room
%    f0 is the starting frequency from which this method is valid.
%    This method is valid in the following frequency range: f0 - 20 kHz
%
%    Room is an array with the coordinates of the room to be used, uses the following format
%                [roomX roomY roomZ]
%    Sources is a matrix with the coordinates of the sources to be used, uses the following format
%                [source1X source1Y source1Z; source2X source2Y source2Z;... ]
%    Mics is a matrix with the coordinates of the microphones to be used, uses the following format
%                [mic1X mic1Y mic1Z; mic2X mic2Y mic2Z;... ]
%    Absorption is an array with the absorption coefficients, uses the following format:
%                 [x-z-walls y-z-walls ceiling floor]
%    fs is the sampling frequency
%    timec is the lenght in seconds for the impulse response calculations.
%    If '0', the program uses the reverberation time, unless the reverberation
%    time is longer than 1 second. In that case, it will make the calculation for 1 second.
%    plotIR indicates if you want to plot the impulse response
%    plotVS indicates if you want to plot the 3D representation of the virtual sources
%
%    Version 0.4.0
%    2007-2008, Carlos Hernandez Matas and Diana Gomez Olmedilla
%    Contact: carlos@hernandezmatas.com and Diana.Gomez.Olmedilla@gmail.com
%
```

This program has been established as an effective estimation of the sound energy in reverberant environments, that can be used by our colleagues in the Virtual Microphones Reseach Group. This way, another strong tool has been developed to be utilized in comparison with other approaches that are being developed in other areas of signal processing

Chapter 5: Results

A pretty rigorous study of the sound propagation in an enclosure space has been done along this thesis. The conclusions achieved after it are that the most outstanding parameters to get a uniform sound distribution are the distance between the source-microphone and matching the walls absorption coefficients of the room.

In our code for the image theory, despite that the second algorithm we developed to calculate the position of the virtual sources –which involved calculating a sphere inside a diamond inside a cube- was fairly more complicated than the first method we designed – calculating just a sphere inside a cube-, it proved to be from 3 to 4 times faster, so finally we chose to use that one.

Due to the increasing computation capabilities, nowadays it's easier to develop algorithms to do more complex and accurate calculus related to architectural acoustics, without having to compromise the time destined to do that calculus, allowing the completion of studies involving complex simulations taking into account a high amount of variables.

Chapter 6: Discussion / Analysis

For future works, some interesting approaches could be those adding more complex features to the method, such as using non-rectangular rooms, adding the effect of furniture or people to the reflections, study the effect of moving sources and/or microphones, etc.

Besides, in terms of the multi-image theory method, it could be interesting to obtain the impulse responses between multiple sources and microphones under an electroacoustic point of view such as the transducers performance in close field or the disturbances created when several transducers are used, instead of the actual method presented here, that is, superposition. Next, make a study to compare the results obtained from both methodologies to check how accurate both techniques are and if the compromise between quickness and accuracy is worth it.

Another interesting study could be to change the specular reflections for diffuse reflections and analyze the difference in the results obtained. However, for that it's needed to analyze the incoming angle of the ray, using complex phases, leading to a much slower algorithm.

Finally, it would be interesting if the reverberation levels are simulated by using recursive filters, as the Schröeder diffuser [15], and compared to our technique.

Also, inside the Virtual Microphones Research Group, we should try to make an effort to combine the different methods about which we are researching, trying to obtain more efficient and precise algorithms for the calculus related to virtual microphones.

References

[1] Lisa Egner, *Architectural Acoustics*, Physics 199POM 12/12/2003

[2] Fletcher, H. and W.A. Munson, *Loudness, its definition, measurement, and calculations*, J.Acous. Soc. Am., 5 (1933) 82-108.

[3] L. Cremer and H.A.Muller, *Principles and applications of Room Acoustics.* Journal of Sound and Vibration, Volume 89, Issue 4, p. 593-596.

[4] F. Alton Everest. *Master Handbook of Acoustics*. Ed. Mc-Graw Hill. 2001

[5] Dr. Y.W. Lam, *Acoustics of Enclosed Spaces* © 1995

[6] F. Pedrotti, L. Pedrotti, *Introduction to Optics (2nd Edition)* Ed. Prentice-Hall 1993

[7] Anna Torelli and Angelo Farina, *Measurement of the sound absorption coefficient of materials with a new sound intensity technique*, Dipartimento Ingegneria Industriale, University of Parma

[8] http://www.ateksis.com/pdf/ses0288.pdf

[9] M.Recuero, *Ingeniería Acústica*, Ed. Paraninfo, 1999

[10] Allen, J. B. and D. A. Berkeley, *Image Method for Efficiently Simulating Small-Room Acoustics*, J. Acous. Soc. Am., Vol. 65(4), pp. 943–950 (April 1979).

[11] L.Kinsler, *Fundamentos de Acústica*, Ed. Limusa, 1988.

[12] http://msis.jsc.nasa.gov/sections/section05.htm

[13] Cremer and Lothar, *Law of the First Wave Front*, JAES Volume 25 Issue 6 pp. 420, 422; June 1977

[14] http://www.sengpielaudio.com/calculator-RT60Coeff.htm

[15] Schroeder, M.R., *New Method of Measuring Reverberation Time*, J. Acous. Soc. Am., 37, 1965, 409-412.

Danilo Simón Zorita, *Apuntes de Acústica*, E.U.I.T.T.

Javier Sánchez Jiménez, *Apuntes de Acústica Arquitéctonica*, E.U.I.T.T

Luis I. Ortiz Berenguer. *Refuerzo sonoro. Bases para el diseño.* 2006

M.Recuero, *Ingeniería Acústica*, Ed. Paraninfo, 1999.

Leo L. Beranek, *Acoustics*, Acoustic Society of America 1993 (last edition)

Anhert, W. y Steffen, F., *Sound reinforcement engineering*, E & FN Spon, London, 1999.

Appendix I [14]

Floor Materials	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
concrete or tile	0.01	0.01	0.15	0.02	0.02	0.02
parquet on concrete	0.04	0.04	0.07	0.06	0.06	0.07
carpet on concrete	0.02	0.06	0.14	0.37	0.60	0.65
carpet on foam	0.08	0.24	0.57	0.69	0.71	0.73

Seating Materials	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
fully occupied - fabric upholstered	0.60	0.74	0.88	0.96	0.93	0.85
empty - fabric upholstered	0.49	0.66	0.80	0.88	0.82	0.70

Wall Materials	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
Brick: unglazed & painted	0.01	0.01	0.02	0.02	0.02	0.03
Concrete block - coarse	0.36	0.44	0.31	0.29	0.39	0.25
Concrete block - painted	0.10	0.05	0.06	0.07	0.09	0.08
Curtain: 18 oz/sq yd fabric molleton	0.14	0.35	0.55	0.72	0.70	0.65
Fiberglass: 2" 703 no airspace	0.22	0.82	0.99	0.99	0.99	0.99
Foam: polyur. 1/2"	0.09	0.11	0.22	0.60	0.88	0.94
Wood: 3/8" plywood panel	0.28	0.22	0.17	0.09	0.10	0.11

Ceiling Materials	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
Acoustic Tiles	0.05	0.22	0.52	0.56	0.45	0.32
Fiberglass: 2" rolls	0.17	0.55	0.80	0.90	0.85	0.80
wood	0.15	0.11	0.10	0.07	0.06	0.07
Foam: Sonex 2"	0.06	0.25	0.56	0.81	0.90	0.91
Plaster: rough on lath	0.02	0.03	0.04	0.05	0.04	0.03
Sheetrock 1/2" 16" on center	0.29	0.10	0.05	0.04	0.07	0.09
Wood: 3/8" plywood panel	0.28	0.22	0.17	0.09	0.10	0.11

Miscellaneous Material	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
Water	0.008	0.008	0.013	0.015	0.020	0.025
People (adults)	0.25	0.35	0.42	0.46	0.5	0.5

Appendix II: imgthry3d.m

function [h, time, tr, f0] = imgthry3d (room, source, mic, absorption, fs, timec, plotIR, plotVS)

```
%
% imgthry3d: Calculates the impulse response of a room using image theory
%
% function [h, time, tr, f0] = imgthry3d (room, source, mic, fs, plotIR, plotVS)
% example of use [h, time, tr, f0] = imgthry3d ([3 4 2.6], [1 1 1], [2 4 1.6], [0.2 0.2 0.2 0.2], 10240, 0, 1, 0);
%
% h is an array with the impulse response of the room for the given mic source and dimensions of the room
% time is the temporal axis of the impulse response
% tr is the reverberation time of the room
% f0 is the starting frequency from which this method is valid.
%    This method is valid in the following frequency range: f0 - 20 kHz
%
% room is an array with the coordinates of the room to be used, uses the following format
%    [roomX roomY roomZ]
% source is an array with the coordinates of the position of the source to be used, uses the following format
%    [sourceX sourceY sourceZ]
% mic is an array with the coordinates of the position of the mic to be used, uses the following format
%    [micX micY micZ]
% absorption is an array with the absorption coefficients, uses the following format:
%    [x-z-walls y-z-walls ceiling floor]
% fs is the sampling frequency
% timec is the lenght in seconds for the impulse response calculations. If
%    '0', the program uses the reverberation time, unless the reverberation
%    time is longer than 1 second. In that case, it will make the calculation
%    for 1 second.
% plotIR indicates if you want to plot the impulse response
% plotVS indicates if you want to plot the 3D representation of the virtual sources
%
% Version 0.4.2
% 2007-2008, Carlos Hernandez Matas and Diana Gomez Olmedilla
% Contact: carlos@hernandezmatas.com and Diana.Gomez.Olmedilla@gmail.com
%

% Constants


X=1;
Y=2;
Z=3;
c = 340.29;
```

```
% Variables

size = 0;

% Calculation of reberveration time

s1 = room(X) * room(Z); % surface of walls parallels to x-z
s2 = room(Y) * room(Z); % surface of walls parallels to y-z
s3 = room(X) * room(Y); % surface of floor + ceiling
rvolume = room(X) * room(Y) * room(Z);

At = 2*s1*absorption(1) + 2*s2*absorption(2) + s3*(absorption(3) + absorption(4));
absorpt = At / (2*(s1 + s2 + s3));

refleccoeff = sqrt(1-absorption);

% R = At / (1 - absorpt);

if absorpt <= 0.2
   tr = (0.16*rvolume) / At; % Sabine
else
   tr = (0.16*rvolume) / (-2*(s1+s2+s3)*log(1-absorpt)); % Eyring
end

if timec == 0
   if tr <= 1
      timec = tr;
   else
      timec = 1;
   end

end

dr = timec*c;

f0 = 45000*sqrt(tr/rvolume);


% Number of images due to reflections

reflections = ceil((1.8)*(dr/min(room)));

if reflections > 0
   for n=1:1:reflections;
      size = size + 4*(n^2)+2;
   end
end
```

```
% Creation of arrays

vsourcex = zeros (size, 1); % stores the virtual source's coordinates
vsourcey = zeros (size,1); % stores the virtual source's coordinates
vsourcez = zeros (size,1); % stores the virtual source's coordinates
vsreflections = zeros (size,3); % stores the amount of reflections for each virtual source
vgrid = zeros ((2*reflections)+1, (2*reflections)+1, (2*reflections)+1); % relates the
geographical location of the images with their position in the list.
gridX = zeros (2, 2 + 2* reflections);
gridY = zeros (2, 2 + 2* reflections);
gridZ = zeros (2, 2 + 2* reflections);


% Calculate the position of the virtual sources, their distance to the real
% mic and the time it takes the sound to arrive.

if (reflections > 0)

   % Positive X Axis

   index=0;
   a = 0;
   b = 1;

   for n=1:1:ceil(dr/min(room))
      if dr >= sqrt ((mic(X)-(2*b*room(X)+(source(X)*(-1)^n)))^2+(mic(Y)-
source(Y))^2+(mic(Z)-source(Z))^2)
         index = index + 1;
         vgrid(reflections+1+n, reflections+1, reflections+1) = index;
         vsourcex(index) = 2 * b * room(X) + (source(X) * (-1)^n);
         vsourcey(index) = source(Y);
         vsourcez(index) = source(Z);
         vsreflections(index, X) = n;
         a = a+1;
         if a == 2
            a = 0;
            b = b+1;
         end
      end
   end


   % Negative X Axis

   if dr >= sqrt ((mic(X)+source(X))^2+(mic(Y)-source(Y))^2+(mic(Z)-source(Z))^2)
      index = index + 1;
      vgrid(reflections, reflections+1, reflections+1) = index;
      vsourcex(index) = -source(X);
```

```
    vsourcey(index) = source(Y);
    vsourcez(index) = source(Z);
    vsreflections(index, X) = 1;
  end

  for n=2:1:ceil(dr/min(room))
    if vgrid(reflections+n, reflections+1, reflections+1) ~= 0
      if dr >= sqrt ((mic(X)+vsourcex(vgrid(reflections+n, reflections+1,
reflections+1)))^2+(mic(Y)-source(Y))^2+(mic(Z)-source(Z))^2)
        index = index + 1;
        vgrid(reflections+1-n, reflections+1, reflections+1) = index;
        vsourcex(index) = -vsourcex(vgrid(reflections+n, reflections+1, reflections+1));
        vsourcey(index) = source(Y);
        vsourcez(index) = source(Z);
        vsreflections(index, X) = n;
      end
    end
  end


  % Positive Y Axis

  a = 0;
  b = 1;

  for n=1:1:ceil(dr/min(room))
    if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-(2*b*room(Y)+(source(Y)*(-
1)^n)))^2+(mic(Z)-source(Z))^2)
      index = index + 1;
      vgrid(reflections+1, reflections+1+n, reflections+1) = index;
      vsourcex(index) = source(X);
      vsourcey(index) = 2 * b * room(Y) + (source(Y) * (-1)^n);
      vsourcez(index) = source(Z);
      vsreflections(index, Y) = n;
      a = a+1;
      if a == 2
        a = 0;
        b = b+1;
      end
    end
  end


  % Negative Y Axis

  if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)+source(Y))^2+(mic(Z)-source(X))^2);
    index = index + 1;
    vgrid(reflections+1, reflections, reflections+1) = index;
    vsourcex(index) = source(X);
```

```
    vsourcey(index) = -source(Y);
    vsourcez(index) = source(Z);
    vsreflections(index, Y) = 1;
  end

  for n=2:1:ceil(dr/min(room))
    if vgrid(reflections+1, reflections+n, reflections+1) ~= 0
      if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)+vsourcey(vgrid(reflections+1,
reflections+n, reflections+1)))^2+(mic(Z)-source(Z))^2);
        index = index + 1;
        vgrid(reflections+1, reflections+1-n, reflections+1) = index;
        vsourcex(index) = source(X);
        vsourcey(index) = - vsourcey(vgrid(reflections+1, reflections+n, reflections+1));
        vsourcez(index) = source(Z);
        vsreflections(index, Y) = n;
      end
    end
  end


  % Positive Z Axis

  a = 0;
  b = 1;

  for n=1:1:ceil(dr/min(room))
    if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-source(Y))^2+(mic(Z)-
(2*b*room(Z)+(source(Z)*(-1)^n)))^2);
      index = index + 1;
      vgrid(reflections+1, reflections+1, reflections+1+n) = index;
      vsourcex(index) = source(X);
      vsourcey(index) = source(Y);
      vsourcez(index) = 2 * b * room(Z) + (source(Z) * (-1)^n);
      vsreflections(index, Z) = n;
      a = a+1;
      if a == 2
        a = 0;
        b = b+1;
      end
    end
  end


  % Negative Z Axis

  if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-source(Y))^2+(mic(Z)+source(Z))^2);
    index = index + 1;
    vgrid(reflections+1, reflections+1, reflections) = index;
    vsourcex(index) = source(X);
```

```
      vsourcey(index) = source(Y);
      vsourcez(index) = -source(Z);
      vsreflections(index, Z) = 1;
    end

    for n=2:1:ceil(dr/min(room))
      if vgrid(reflections+1, reflections+1, reflections+n) ~= 0
        if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-
source(Y))^2+(mic(Z)+vsourcez(vgrid(reflections+1, reflections+1, reflections+n)))^2);
          index = index + 1;
          vgrid(reflections+1, reflections+1, reflections+1-n) = index;
          vsourcex(index) = source(X);
          vsourcey(index) = source(Y);
          vsourcez(index) = - vsourcez(vgrid(reflections+1, reflections+1, reflections+n));
          vsreflections(index, Z) = n;
        end
      end
    end


    % +X +Y, +X +Z, +Y +Z, +X +Y +Z, +X +Y -Z

    for l=(reflections+2):1:((2*reflections) + 1)
      for k=(reflections+2):1:((2*reflections) + 1)
        if ((k + l - 2*(reflections+1)) <= reflections)

          % +X +Y
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
source(Z))^2)
                index = index+1;
                vgrid(k, l, reflections+1)= index;
                vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                vsourcez(index) = source(Z);
                vsreflections(index, X) = abs(k-(reflections+1));
                vsreflections(index, Y) = abs(l-(reflections+1));
              end
            end
          end

          % +X +Z
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, reflections+1, l) ~= 0
              if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-source(Y))^2+(mic(Z)-vsourcez(vgrid(reflections+1,
reflections+1, l)))^2)
```

```
                index = index+1;
                vgrid(k, reflections+1, l)= index;
                vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                vsourcey(index) = source(Y);
                vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
                vsreflections(index, X) = abs(k-(reflections+1));
                vsreflections(index, Z) = abs(l-(reflections+1));
              end
            end
          end

          % +Y +Z
          if vgrid(reflections+1, k, reflections+1) ~= 0
            if vgrid(reflections+1, reflections+1, l) ~= 0
              if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-vsourcey(vgrid(reflections+1, k,
reflections+1)))^2+(mic(Z)-vsourcez(vgrid(reflections+1, reflections+1, l)))^2)
                index = index+1;
                vgrid(reflections+1, k, l)= index;
                vsourcex(index) = source(X);
                vsourcey(index) = vsourcey(vgrid(reflections+1, k, reflections+1));
                vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
                vsreflections(index, Y) = abs(k-(reflections+1));
                vsreflections(index, Z) = abs(l-(reflections+1));
              end
            end
          end
        end


        % +X +Y +Z
        for m=(reflections+2):1:((2*reflections) + 1)
          if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
            if vgrid(k, reflections+1, reflections+1) ~= 0
              if vgrid(reflections+1, l, reflections+1) ~= 0
                if vgrid(reflections+1, reflections+1, m) ~= 0
                  if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                    index = index+1;
                    vgrid(k, l, m)= index;
                    vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                    vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                    vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                    vsreflections(index, X) = abs(k-(reflections+1));
                    vsreflections(index, Y) = abs(l-(reflections+1));
                    vsreflections(index, Z) = abs(m-(reflections+1));
                  end
                end
```

```
            end
          end
        end
      end

      % +X +Y -Z
      for m=1:1:reflections
        if ((k + l + abs(m-(reflections+1)) - 2*(reflections+1)) <= reflections)
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if vgrid(reflections+1, reflections+1, m) ~= 0
                if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                  index = index+1;
                  vgrid(k, l, m)= index;
                  vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                  vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                  vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                  vsreflections(index, X) = abs(k-(reflections+1));
                  vsreflections(index, Y) = abs(l-(reflections+1));
                  vsreflections(index, Z) = abs(m-(reflections+1));
                end
              end
            end
          end
        end
      end
    end
  end


  % +X -Y, +X -Z, +Y -Z, +X -Y +Z, +X -Y -Z

  for l=1:1:reflections
    for k=(reflections+2):1:((2*reflections) + 1)
      if (l >= (k-reflections))

        % +X -Y
        if vgrid(k, reflections+1, reflections+1) ~= 0
          if vgrid(reflections+1, l, reflections+1) ~= 0
            if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
source(Z))^2)
              index = index+1;
              vgrid(k, l, reflections+1)= index;
              vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
              vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
              vsourcez(index) = source(Z);
```

```
                vsreflections(index, X) = abs(k-(reflections+1));
                vsreflections(index, Y) = abs(l-(reflections+1));
              end
            end
          end

          % +X -Z
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, reflections+1, l) ~= 0
              if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-source(Y))^2+(mic(Z)-vsourcez(vgrid(reflections+1,
reflections+1, l)))^2)
                index = index+1;
                vgrid(k, reflections+1, l)= index;
                vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                vsourcey(index) = source(Y);
                vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
                vsreflections(index, X) = abs(k-(reflections+1));
                vsreflections(index, Z) = abs(l-(reflections+1));
              end
            end
          end

          % +Y -Z
          if vgrid(reflections+1, k, reflections+1) ~= 0
            if vgrid(reflections+1, reflections+1, l) ~= 0
              if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-vsourcey(vgrid(reflections+1, k,
reflections+1)))^2+(mic(Z)-vsourcez(vgrid(reflections+1, reflections+1, l)))^2)
                index = index+1;
                vgrid(reflections+1, k, l)= index;
                vsourcex(index) = source(X);
                vsourcey(index) = vsourcey(vgrid(reflections+1, k, reflections+1));
                vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
                vsreflections(index, Y) = abs(k-(reflections+1));
                vsreflections(index, Z) = abs(l-(reflections+1));
              end
            end
          end
        end

        % +X -Y +Z
        for m=(reflections+2):1:((2*reflections) + 1)
          if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
            if vgrid(k, reflections+1, reflections+1) ~= 0
              if vgrid(reflections+1, l, reflections+1) ~= 0
                if vgrid(reflections+1, reflections+1, m) ~= 0
```

```
                    if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                      index = index+1;
                      vgrid(k, l, m)= index;
                      vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                      vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                      vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                      vsreflections(index, X) = abs(k-(reflections+1));
                      vsreflections(index, Y) = abs(l-(reflections+1));
                      vsreflections(index, Z) = abs(m-(reflections+1));
                    end
                  end
                end
              end
            end
          end

          % +X -Y -Z
          for m=1:1:reflections
            if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
              if vgrid(k, reflections+1, reflections+1) ~= 0
                if vgrid(reflections+1, l, reflections+1) ~= 0
                  if vgrid(reflections+1, reflections+1, m) ~= 0
                    if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                      index = index+1;
                      vgrid(k, l, m)= index;
                      vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                      vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                      vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                      vsreflections(index, X) = abs(k-(reflections+1));
                      vsreflections(index, Y) = abs(l-(reflections+1));
                      vsreflections(index, Z) = abs(m-(reflections+1));
                    end
                  end
                end
              end
            end
          end
        end
      end
    end

    % -X -Y, -X -Z, -Y -Z, -X -Y +Z, -X -Y -Z

    for l=1:1:reflections
```

```
    for k=1:1:reflections
      if (k + l -1 > reflections)

        % -X -Y
        if vgrid(k, reflections+1, reflections+1) ~= 0
          if vgrid(reflections+1, l, reflections+1) ~= 0
            if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
source(Z))^2)
              index = index+1;
              vgrid(k, l, reflections+1)= index;
              vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
              vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
              vsourcez(index) = source(Z);
              vsreflections(index, X) = abs(k-(reflections+1));
              vsreflections(index, Y) = abs(l-(reflections+1));
            end
          end
        end

        % -X -Z
        if vgrid(k, reflections+1, reflections+1) ~= 0
          if vgrid(reflections+1, reflections+1, l) ~= 0
            if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-source(Y))^2+(mic(Z)-vsourcez(vgrid(reflections+1,
reflections+1, l)))^2)
              index = index+1;
              vgrid(k, reflections+1, l)= index;
              vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
              vsourcey(index) = source(Y);
              vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
              vsreflections(index, X) = abs(k-(reflections+1));
              vsreflections(index, Z) = abs(l-(reflections+1));
            end
          end
        end

        % -Y -Z
        if vgrid(reflections+1, k, reflections+1) ~= 0
          if vgrid(reflections+1, reflections+1, l) ~= 0
            if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-vsourcey(vgrid(reflections+1, k,
reflections+1)))^2+(mic(Z)-vsourcez(vgrid(reflections+1, reflections+1, l)))^2)
              index = index+1;
              vgrid(reflections+1, k, l)= index;
              vsourcex(index) = source(X);
              vsourcey(index) = vsourcey(vgrid(reflections+1, k, reflections+1));
              vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
              vsreflections(index, Y) = abs(k-(reflections+1));
              vsreflections(index, Z) = abs(l-(reflections+1));
```

```
            end
          end
        end
      end

      % -X -Y +Z
      for m=(reflections+2):1:((2*reflections) + 1)
        if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if vgrid(reflections+1, reflections+1, m) ~= 0
                if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                  index = index+1;
                  vgrid(k, l, m)= index;
                  vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                  vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                  vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                  vsreflections(index, X) = abs(k-(reflections+1));
                  vsreflections(index, Y) = abs(l-(reflections+1));
                  vsreflections(index, Z) = abs(m-(reflections+1));
                end
              end
            end
          end
        end

      % -X -Y -Z
      for m=1:1:reflections
        if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if vgrid(reflections+1, reflections+1, m) ~= 0
                if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                  index = index+1;
                  vgrid(k, l, m)= index;
                  vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                  vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                  vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                  vsreflections(index, X) = abs(k-(reflections+1));
                  vsreflections(index, Y) = abs(l-(reflections+1));
                  vsreflections(index, Z) = abs(m-(reflections+1));
                end
```

```
                end
            end
        end
    end
  end
end


% -X +Y, -X +Z, -Y +Z, -X +Y +Z, -X +Y -Z

for l=(reflections+2):1:((2*reflections) + 1)
  for k=1:1:reflections
    if ((l-reflections) <= k)

      % -X +Y
      if vgrid(k, reflections+1, reflections+1) ~= 0
        if vgrid(reflections+1, l, reflections+1) ~= 0
          if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
source(Z))^2)
            index = index+1;
            vgrid(k, l, reflections+1)= index;
            vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
            vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
            vsourcez(index) = source(Z);
            vsreflections(index, X) = abs(k-(reflections+1));
            vsreflections(index, Y) = abs(l-(reflections+1));
          end
        end
      end

      % -X +Z
      if vgrid(k, reflections+1, reflections+1) ~= 0
        if vgrid(reflections+1, reflections+1, l) ~= 0
          if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-source(Y))^2+(mic(Z)-vsourcez(vgrid(reflections+1,
reflections+1, l)))^2)
            index = index+1;
            vgrid(k, reflections+1, l)= index;
            vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
            vsourcey(index) = source(Y);
            vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
            vsreflections(index, X) = abs(k-(reflections+1));
            vsreflections(index, Z) = abs(l-(reflections+1));
          end
        end
      end
```

```
        % -Y +Z
        if vgrid(reflections+1, k, reflections+1) ~= 0
          if vgrid(reflections+1, reflections+1, l) ~= 0
            if dr >= sqrt ((mic(X)-source(X))^2+(mic(Y)-vsourcey(vgrid(reflections+1, k,
reflections+1)))^2+(mic(Z)-vsourcez(vgrid(reflections+1, reflections+1, l)))^2)
              index = index+1;
              vgrid(reflections+1, k, l)= index;
              vsourcex(index) = source(X);
              vsourcey(index) = vsourcey(vgrid(reflections+1, k, reflections+1));
              vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, l));
              vsreflections(index, Y) = abs(k-(reflections+1));
              vsreflections(index, Z) = abs(l-(reflections+1));
            end
          end
        end
      end

      % -X +Y +Z
      for m=(reflections+2):1:((2*reflections) + 1)
        if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if vgrid(reflections+1, reflections+1, m) ~= 0
                if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                  index = index+1;
                  vgrid(k, l, m)= index;
                  vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                  vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                  vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                  vsreflections(index, X) = abs(k-(reflections+1));
                  vsreflections(index, Y) = abs(l-(reflections+1));
                  vsreflections(index, Z) = abs(m-(reflections+1));
                end
              end
            end
          end
        end
      end

      % -X +Y -Z
      for m=1:1:reflections
        if ((abs(k -(reflections+1)) + abs(l -(reflections+1)) + abs(m -(reflections+1))) <=
reflections)
          if vgrid(k, reflections+1, reflections+1) ~= 0
            if vgrid(reflections+1, l, reflections+1) ~= 0
              if vgrid(reflections+1, reflections+1, m) ~= 0
```

```
                    if dr >= sqrt ((mic(X)-vsourcex(vgrid(k, reflections+1,
reflections+1)))^2+(mic(Y)-vsourcey(vgrid(reflections+1, l, reflections+1)))^2+(mic(Z)-
vsourcez(vgrid(reflections+1, reflections+1, m)))^2)
                      index = index+1;
                      vgrid(k, l, m)= index;
                      vsourcex(index) = vsourcex(vgrid(k, reflections+1, reflections+1));
                      vsourcey(index) = vsourcey(vgrid(reflections+1, l, reflections+1));
                      vsourcez(index) = vsourcez(vgrid(reflections+1, reflections+1, m));
                      vsreflections(index, X) = abs(k-(reflections+1));
                      vsreflections(index, Y) = abs(l-(reflections+1));
                      vsreflections(index, Z) = abs(m-(reflections+1));
                    end
                  end
                end
              end
            end
          end
        end
      end

end


vsdistance=sqrt((vsourcex - mic(X)).^2+(vsourcey - mic(Y)).^2+(vsourcez - mic(Z)).^2);
samples = round(fs*vsdistance/c);

IR = zeros (ceil(timec*fs),1);


% Doaks & Bolt

for n=1:1:index
  if room(Z) < vsourcez(n)
    vsreflectionsc = ceil(vsreflections(n, Z)/2);
    vsreflectionsf = floor(vsreflections(n, Z)/2);
  else
    vsreflectionsc = floor(vsreflections(n, Z)/2);
    vsreflectionsf = ceil(vsreflections(n, Z)/2);
  end
  IR(samples(n))=IR(samples(n))+((1*(refleccoeff(1)^vsreflections(n,
X))*(refleccoeff(2)^vsreflections(n,
Y))*(refleccoeff(3)^vsreflectionsc)*(refleccoeff(4)^vsreflectionsf))/(vsdistance(n)));
end


% % alternate the sign of the virtual sources
%
% for n=1:1:index
%   if mic(Z) < vsourcez(n)
```

```
%       vsreflectionsc = ceil(vsreflections(n, Z)/2);
%       vsreflectionsf = floor(vsreflections(n, Z)/2);
%    else
%       vsreflectionsc = floor(vsreflections(n, Z)/2);
%       vsreflectionsf = ceil(vsreflections(n, Z)/2);
%    end
%    IR(samples(n))=IR(samples(n))+(((1*(refleccoeff(1)^vsreflections(n,
X))*(refleccoeff(2)^vsreflections(n,
Y))*(refleccoeff(3)^vsreflectionsc)*(refleccoeff(4)^vsreflectionsf))/(vsdistance(n)))*sign
(rand(1)*2-1));
% end


% add the effect of the direct ray

drdistance = sqrt (((mic(X)-source(X))^2)+((mic(Y)-source(Y))^2)+((mic(Z)-
source(Z))^2));
t = drdistance/c;
IR(round(t*fs)) = 1 / drdistance;

h = IR/max(abs(IR)); % normalize the signal to 1

time = 0:(1/fs):((length(IR)-1)/fs);

if plotIR
   figure
   bar (time, h);
   axis tight
   title('Impulse Response')
   xlabel('Time (s)')
   ylabel('Normalized Amplitude')
end


% Plot the graph

if plotVS == 1

   % Calulate the lines

   for n=1:1:(2 + 2* ceil(dr/min(room)))
      gridX (1,n) = ((room(X)*(n-1)) - (ceil(dr/min(room))*room(X)));
      gridX (2,n) = ((room(X)*(n-1)) - (ceil(dr/min(room))*room(X)));
      gridY (1,n) = ((room(Y)*(n-1)) - (ceil(dr/min(room))*room(Y)));
      gridY (2,n) = ((room(Y)*(n-1)) - (ceil(dr/min(room))*room(Y)));
      gridZ (1,n) = ((room(Z)*(n-1)) - (ceil(dr/min(room))*room(Z)));
      gridZ (2,n) = ((room(Z)*(n-1)) - (ceil(dr/min(room))*room(Z)));
   end
```

```
axX = [gridX(1,1) gridX(1, (2 + 2* ceil(dr/min(room))))];
axY = [gridY(1,1) gridY(1, (2 + 2* ceil(dr/min(room))))];
axZ = [gridZ(1,1) gridZ(1, (2 + 2* ceil(dr/min(room))))];

auxX = [gridX(1, (2 + 2* ceil(dr/min(room)))) gridX(1, (2 + 2* ceil(dr/min(room))))];
auxY = [gridY(1, (2 + 2* ceil(dr/min(room)))) gridY(1, (2 + 2* ceil(dr/min(room))))];
auxZ = [gridZ(1,1) gridZ(1,1)];

% Plot the points

figure
plot3 (mic(X), mic(Y), mic(Z),'k*') % Plots the location of the mic
hold on
plot3 (source(X), source(Y), source(Z), 'ko') % Plots the location of the real source

plot3([mic(X) source(X)],[mic(Y) source(Y)],[mic(Z) source(Z)], 'b-') % Plots a line
between the mic and the real source

if reflections > 0
   for n=1:1:index
      plot3 (vsourcex(n), vsourcey(n), vsourcez(n), 'bo') % Plots the virtual sources
      plot3 ([mic(X) vsourcex(n)],[mic(Y) vsourcey(n)],[mic(Z) vsourcez(n)],'m--') % Plot
a line between the mic and te virtual sources

   end
end

% X-Y plane grid

plot3 (gridX, axY, auxZ, 'k-')
plot3 (axX, gridY, auxZ,'k-')

% X-Z plane grid

plot3 (gridX, auxY, axZ, 'k-')
plot3 (axX, auxY, gridZ,'k-')

% Y-Z plane grid

plot3 (auxX, gridY, axZ, 'k-')
plot3 (auxX, axY, gridZ,'k-')

[as,bs,cs]=sphere;
as=(dr*as)+mic(X);
bs=(dr*bs)+mic(Y);
cs=(dr*cs)+mic(Z);
plot3(as,bs,cs,'k-');
plot3(cs,bs,as,'k-');
```

```
    axis equal
    axis tight
    title('Image Theory 3D')
    xlabel('X')
    ylabel('Y')
    zlabel('Z')

end
```

Appendix III: multi_imgthry3d.m

function [h, time, tr, f0] = multi_imgthry3d (room, sources, mics, absorption, fs, timec, plotIR, plotVS)

```
%
% multi_imgthry3d: Applies imgthry3d to several mics and sources
%
% function [h, time, tr, f0] = multi_imgthry3d (room, sources, mics, absorption, fs, plotIR, plotVS)
% example of use: [h, time, tr, f0] = multi_imgthry3d ([3 4 2.6], [1 1 1; 1 1 2], [2 2.5 2; 2 4 1.6], [0.2 0.2 0.2 0.2], 44100, 0, 1, 0);
%
% h is a matrix with the impulse responses of every combination of sources
%   and mics following this order:
%   source1mic1
%   source1mic2
%   ...
%   source1micN
%   ...
%   sourceMmicN
% time is the temporal axis of the impulse responses
% tr is the reverberation time of the room
% f0 is the starting frequency from which this method is valid.
%    This method is valid in the following frequency range: f0 - 20 kHz
%
% room is an array with the coordinates of the room to be used, uses the following format
%    [roomX roomY roomZ]
% sources is a matrix with the coordinates of the sources to be used, uses the following format
%    [source1X source1Y source1Z; source2X source2Y source2Z;... ]
% mics is a matrix with the coordinates of the mics to be used, uses the following format
%    [mic1X mic1Y mic1Z; mic2X mic2Y mic2Z;... ]
% absorption is an array with the absorption coefficients, uses the following format:
%    [x-z-walls y-z-walls ceiling floor]
% fs is the sampling frequency
% timec is the lenght in seconds for the impulse response calculations. If
%    '0', the program uses the reverberation time, unless the reverberation
%    time is longer than 1 second. In that case, it will make the calculation
%    for 1 second.
% plotIR indicates if you want to plot the impulse response
% plotVS indicates if you want to plot the 3D representation of the virtual sources
%
% Version 0.4.0
% 2007-2008, Carlos Hernandez Matas and Diana Gomez Olmedilla
% Contact: carlos@hernandezmatas.com and Diana.Gomez.Olmedilla@gmail.com
%
```

```
% Constants

X=1;
Y=2;
Z=3;

a = (numel(sources)/3);
b = (numel(mics)/3);


% Calculation of reberveration time

s1 = room(X) * room(Z); % surface of walls parallels to x-z
s2 = room(Y) * room(Z); % surface of walls parallels to y-z
s3 = room(X) * room(Y); % surface of floor + ceiling
rvolume = room(X) * room(Y) * room(Z);

At = 2*s1*absorption(1) + 2*s2*absorption(2) + s3*(absorption(3) + absorption(4));
absorpt = At / (2*(s1 + s2 + s3));


if absorpt <= 0.2
   tr = (0.16*rvolume) / At; % Sabine
else
   tr = (0.16*rvolume) / (-2*(s1+s2+s3)*log(1-absorpt)); % Eyring
end

if timec == 0
   if tr <= 1
      timec = tr;
   else
      timec = 1;
   end

end

h = zeros(a*b, ceil(timec*fs));


index = 1;

for k=1:1:(numel(sources)/3)
   for l=1:1:(numel(mics)/3)
      [h(index,:), time, tr, f0] = imgthry3d(room, sources(k,:), mics(l,:), absorption, fs, timec,
plotIR, plotVS);
      index = index+1;
   end
end
```