Physics-Based Modelling of Musical Instruments

Assignment 3: Image Source Reverberation

The deadline for this assignment is Monday 29th February 2016 by 11.59pm.

Assignment structure

This assignment has two main parts for the basic coding:

1. Design a script that generates an impulse response, stored in a `.wav` file called '`ir_roomsize...wav`', using the image source method. [`'ImageSource...m'`]

2. Design a script that convolves an input `.wav` file of your choice with the impulse response you have simulated using the '`ImageSource...m`' script, in order to make the sound in the input `.wav` file appear to have been created and recorded in a virtual room. [`'ConvReverb...m'`]

Submission details

- Please submit at least *four files* for the basic assignment. These will include the two '`.m`' program files described in the assignment, plus the `.wav` file containing your simulated impulse response, plus the `.wav` sound file you have been using as an input (a selection of sample sound files are on the Learn website). The files should be named as:

  1. `ImageSource_StudentNumber_Surname_Firstname.m`
  2. `ConvReverb_StudentNumber_Surname_Firstname.m`
  3. `IR_roomsize_StudentNumber_Surname_Firstname.wav`
  4. `wavefilename.wav`

  where '`StudentNumber`', '`Surname`', '`Firstname`' and '`wavefilename`' should be replaced as appropriate, and '`roomsize`' should be some relevant text/numbers to indicate the size of the room you have simulated. If you choose to include further work/programs, please do so using a similar filenaming convention.

- Submission is via the Learn website for the course. Go into the 'Assignments' folder and upload from there.

  – Make sure you upload your completed assignment files to the appropriate assignment.
  – Make sure files are named appropriately.
  – You can replace uploaded files with newer versions if you want to, up until the deadline.
  – The Learn platform will automatically carry out a plagiarism check on submitted files.

Resources

- J. A. Moorer, "About this reverberation business", Computer Music Journal 3(2), pp. 13-18, 1979.

- Jont B. Allen and David A. Berkley, "Image method for efficiently simulating small-room acoustics", JASA 65(4), 1979.

- E. A. Lehmann and A. M. Johansson, "Diffuse reverberation model for efficient image-source simulation of room impulse responses", IEEE Transactions on Audio, Speech and Language Processing 18(6), 2010.

  – All papers are available on the Learn website for this course.

Marking criteria

- A *completely perfect* basic assignment, completed as outlined in this document, can typically gain up to 67% for the UG version of the course, and up to 60% for the PG version of the course.
- In order to obtain higher marks, you must seek out ways to code more interesting and in depth programs around the theme of the assignment. There are some suggestions at the end of the document under 'Beyond the basics' that provide a starting point, and other ideas can often be found in the published literature (see 'Resources'). Note that simply creating a GUI of the basic assignment might not gain as many marks as researching and coding a more in depth program.

3.1 Image source reverberation

Introduction

The image source method is basically a very accurate way of modelling the reverberation in a room of simple shape. We effectively assume that the walls act as mirrors for the acoustic waves in the room in that the angle of reflection is equal to the angle of incidence. For a three dimensional room with rectangular sides this means that, acoustically speaking, we can see a source of sound repeated an infinite number of times throughout space. Figure 1 shows a two dimensional version of this:

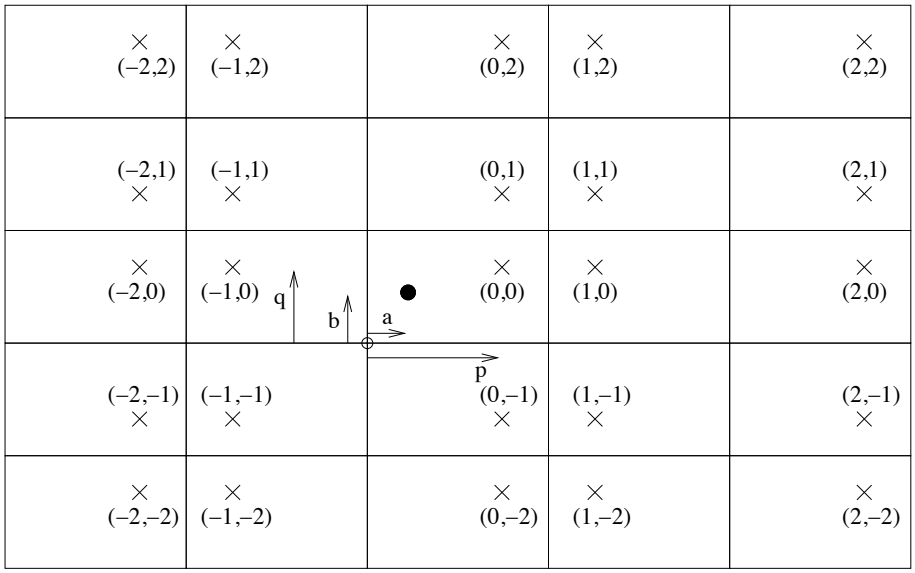

Figure 1: Image source method for a rectangular room

The listener is represented by the black circle at a position of $x = a$, $y = b$ while the source is the cross located at $x = p$, $y = q$ and labelled $(0, 0)$. Virtual sources are found reflected in the walls of the room and these are labelled with the first index showing the number of the room along the x axis and the second index showing the number of the virtual room along the y axis.

We will assume that the source and all the virtual sources emit an impulse simultaneously and calculate the distances travelled to the listener and from this work out the times of arrival at the listener and how the amplitude of the pulse has changed. Note that the path from each of the virtual sources to the listener is equivalent in length to a path between the actual source and listener consisting of a number of reflections within the room. The first index in each bracket indicates how many times the source hits a wall before arriving at the listener while the second index indicates how many times the impulse hits the ceiling and/or floor. Every time an impulse arrival time and magnitude has been calculated we can combine this information together, placing it into the simulated impulse response

vector, by adding the appropriate magnitude at the sample number appropriate for the corresponding arrival time.

Let us consider now what the assumptions in the model mean. For reflection to occur at equal angles the size of the reflecting surfaces must be much larger than the wavelength of the reflecting sound. In other words specular reflections happen rather than the diffuse reflections which happen to some extent in all real-world situations when sound waves diffract round or from objects of small size. While we are assuming that the walls are acoustic mirrors, we must take into account that energy will be absorbed at each reflection so we include an absorption coefficient, α. In real life the absorption coefficient will be frequency dependent, with the absorption increasing with increasing frequency. For our model to be computationally efficient, however, we will initially take the approximation that the absorption coefficient will be frequency independent. The reverb model will sound beautifully complex, but a little too harsh as the sound decays.

Computing a simulated impulse response

Calculating the arrival time of each virtual reflected impulse

Consider the distance between the listener and a virtual source in a virtual room labelled (d, e, f) in a 3d version of the grid in figure 1. By Pythagoras the distance is:

$$l_{d,e,f} = \sqrt{A_d^2 + B_e^2 + C_f^2}$$

where

$$A_d = \begin{cases} (d+1)L_x - p - a & : \quad d \text{ odd} \\ dL_x + p - a & : \quad d \text{ even,} \end{cases}$$

$$B_e = \begin{cases} (e+1)L_y - q - b & : \quad e \text{ odd} \\ eL_y + q - b & : \quad e \text{ even} \end{cases}$$

and

$$C_f = \begin{cases} (f+1)L_z - r - c & : \quad f \text{ odd} \\ fL_z + r - c & : \quad f \text{ even} \end{cases}$$

where L_x, L_y and L_z are the widths of the room in the x, y and z dimension respectively, c is the position of the listener in the z direction and r is the position of the source in the z direction. Chose realistic non-integer multiples for these dimensions. The first task is thus to calculate these lengths using three nested for loops which step through $d = -N : N$, $e = -N : N$ and $f = -N : N$ respectively where $N = 10$ might be a sensible choice for typical geometries. The distance of travel ($l_{d,e,f}$) is calculated, then the time of arrival using

$$t = \frac{l_{d,e,f}}{343}.$$

Note that we can estimate a time duration, tf, that is certain to include **all** reflections for a given order of N by calculating the approximate reflection time for an 'extra' layer of virtual rooms

$$tf = \frac{\sqrt{[(N+1)L_x]^2 + [(N+1)L_y]^2 + [(N+1)L_z]^2}}{343}.$$

Thus at the start of the code an 'empty' impulse response vector with an appropriate length may be initialised, which will be guaranteed to be (just slightly) longer than needed to contain all the virtual room reflections. Note that this will result in an impulse response vector h with zeros at the end. It is left as a 'Beyond the basics' exercise for you to precisely calculate the time of the last impulse response signal in advance of the main loop (i.e. to improve the efficiency of the given method).

Calculating the amplitude of each virtual reflected impulse

The magnitude of the each impulse is given by

$$g = (1/l_{d,e,f}) * \alpha^w$$

where α is the acoustic absorption of the walls (you will need to try numbers in a scale from 0 to 1) and w is the number of collisions between the walls and the sound wave (which is also left as an exercise for you to work out). Next the sample number corresponding to the time of arrival is calculated and the results put into a vector of the impulse response. The impulse response vector should look like an exponentially decaying set of positive impulses arriving with a greater density with increasing time. Sometimes impulses from different routes arrive in the same time step and the amplitude seems to double. Don't worry about this as it is just a natural consequence of time quantisation.

Exporting the virtual impulse response signal

Write your impulse response vector as a wav file which begins with the characters 'ir_roomsize' and includes a note of the room size you used (i.e. replace 'roomsize' with some suitable text/numbers to indicate the parameters you have used; this should happen automatically so if the parameters change, the output wav filename will change).

- In your submitted assignment, the default room size in your '`ImageSource...m`' program script must be set to be 10m x 10m x 10m, and you should use an order value N of 10. Your default value of α must be 0.9. Your source should be located at (x, y, z) coordinates of (2, 7, 1), and your receiver at coordinates of $(2, 4, 5)$. Your output simulated impulse response, as stored in the `IR_roomsize_StudentNumber_Surname_Firstname.wav` wave file, should be encoded at a bit depth of 16bits, and a sample rate of 44.1kHz.

3.2 Convolution reverb

In order to implement the reverb and hear what music sounds like in the virtual space we have created, create a script (`ConvReverb_StudentNumber_Surname_Firstname.m`) to perform a convolution reverb operation. First read in a `.wav` file of suitable audio, then read in the `.wav` file of the simulated impulse response, and then convolve the two. The results should be written as a new, output `.wav` file. The script should be robust, including any error checking that you deem appropriate.

The convolution can be done using a vector based implimentation as the impulse response vector simply consists of feed forward coefficients (there is no feedback). There are a number of ways to actually implement this convolution in practice, both in the time domain and the frequency domain. Credit will be given if you demonstrate how to code the convolution in the time domain by hand, in the basic version of the assignment. You may wish to explore this issue further, including comparisons between methods, as part of any 'Beyond the basics' work.

Beyond the basics

To further improve the code you may like to include a way of giving different walls in the room a different absorption coefficient. The computation can also be altered to model frequency dependent absorption coefficients. You might do this by defining a different absorption coefficient for each of a number of frequency 'octave bands', and running a separate ISR model for each one, before combining the results. Good coding/explanation of this will gain significant extra credit.

You might also like to consider how stereo reverberation could be efficiently implemented using this type of scheme. Can a directional factor, applied as part of the receiver positioning and readout, be used? As a first attempt, you might explore using two different listening positions to simulate time delays for sound travelling different paths to each of two ears. There is no easy way of including the effects of diffraction around the listener's head in all of this so strictly speaking the resulting sounds will sound most accurate if listened to on loudspeakers placed at suitable receiver location(s) in a room with the same dimensions/absorption as your model. The assessment here, however, is of your coded programs.

You could also experiment with faster implementations of the convolution procedure. Particular credit will be given if you can demonstrate sample-for-sample equivalence of, say, alternative/equivalent time and frequency domain methods.

All work going beyond the basic coding should be saved in appropriately named new scripts/programs.