Master MVA: Sound Rendering (TP)

Gaël RICHARD

17 février 2020

1 First part

1.1 Introduction

In this first part, we will listen to some binaural rendering examples using the MATLAB programs *show_data* and *hor_show_data* distributed by the University of California.

The MATLAB script program *show_data* allows to visualize the HRTF (Head Related Transfer Function) database recorded by the CIPIC Interface Laboratory of University of California.

The graphical interface allows in particular:

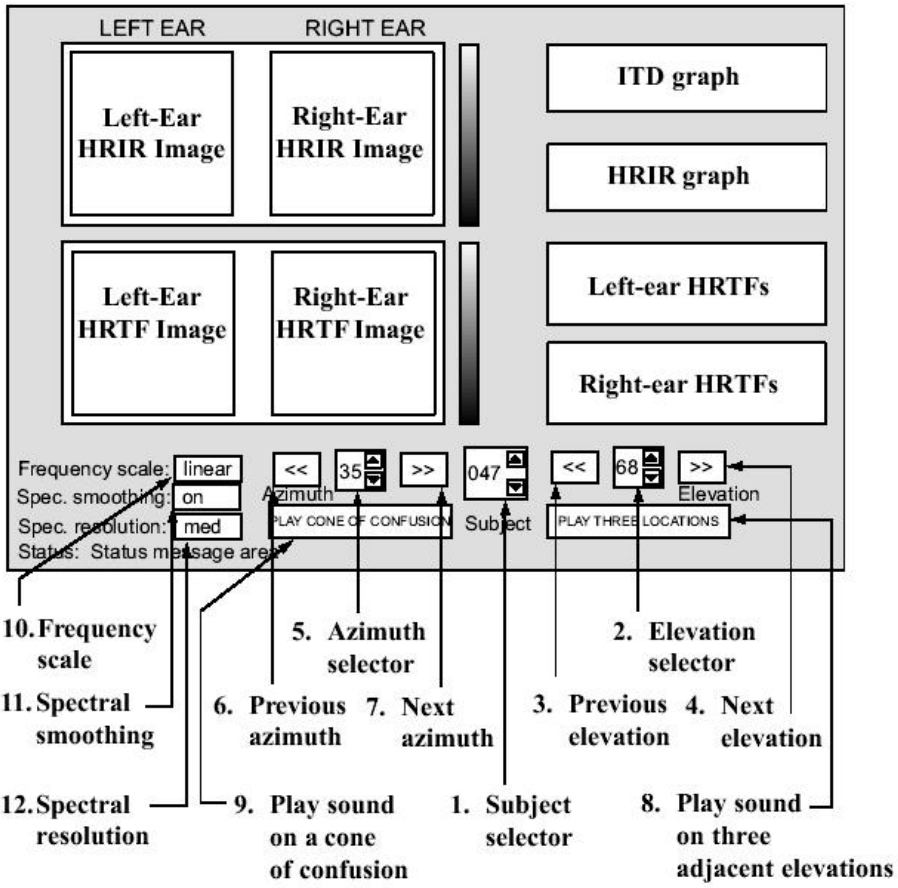
- to load a file containing the HRTF for a given subject
- to visualize simultaneously the impulse responses and frequency responses of HRTFs for the left and right ears.
- to change azimuth and elevation
- to listen spatialised sounds with a given azimuth/elevation or a sequence of sounds with elevation taken on the *cone of confusion*.

The other program, *hor_show_data*, is very similar and allows to visualize HRTFS or listen to spatialised sounds in an horizontal plane.

1.2 How to use the programs *show_data* and *hor_show_data*

The process to use the scripts is :

- Launch MATLAB
- Go to directory **show_data**
- Type *show_data* at the Matlab prompt. A graphical interface should appear (see figure 1.2)
- Load the HRTF data in selecting a subject number (note that only subject numbers 003 and 065 are available!!)



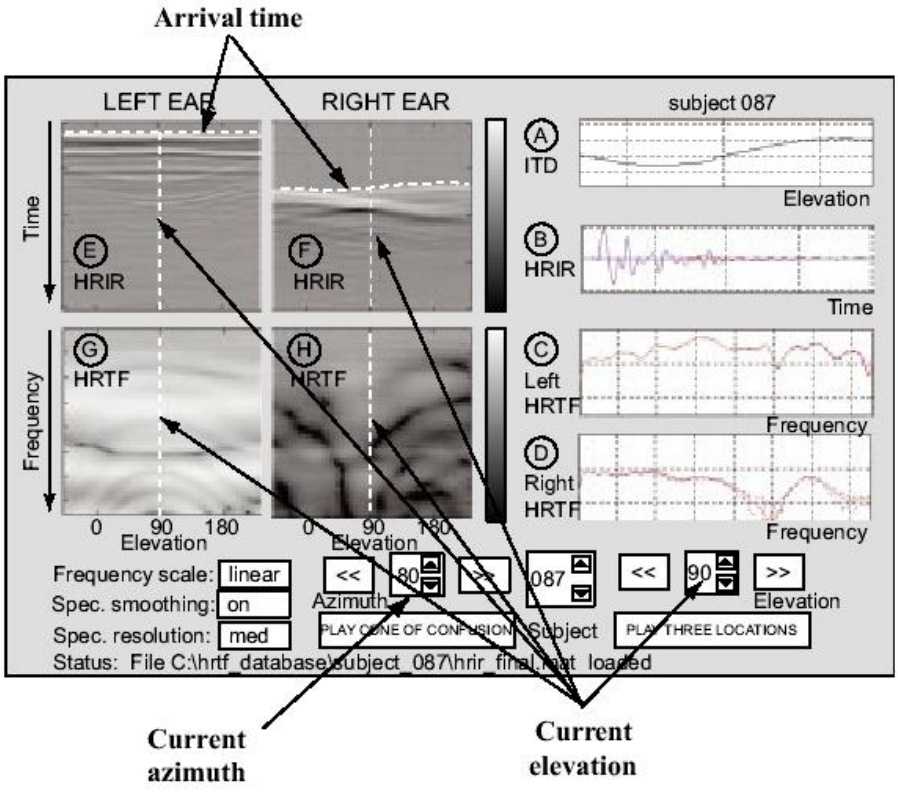


FIG. 1 – *Graphical interface of MATLAB program* show_data

Then, you can :

- Select a point in space (azimuth/elevation) and play sounds at three positions which are close in space (press on *Play Three Locations*)
- Play sounds at 50 different positions regularly placed on the cone of confusion (press : *Play cone of confusion*)
- Compare results by selecting another dataset of HRTFs

You can also run and do similar operations with the program *hor_show_data*.

2 Second part

In this part, we propose to program a spatializer for binaural listening (e.g. using headphones). To that aim, we will use a dataset of HRTFs or more precisely HRIR (Head Related Impulse responses). Each HRIR is sampled at 44.1 kHz and depends on azimuth, elevation and time. In the datafile, HRIRs are stored in two arrays containing all HRIRs recorded at discrete positions in space. Then, the azimuth, elevation and time values are respectively specified by the discrete indices *naz*, *nel* and *nt*. (*naz*,*nel*,*nt*) is then an array of $25 \times 50 \times 200$ (25 azimuths, 50 elevations and 200 time samples).

The azimuth is an angle θ defined between the mid-sagittal plane and the vertical plane containing the head center and the considered position. Values are then between -90 and $+90$ (Values are not equally spread since there are a greater density of points between -45 and $+45$).

The elevation is an angle defined between the horizontal plane and the plane containing the source and the head center. Values are given by steps of 5.625 between -45 and $+230$, knowing that there are no values between -90 and -45 and between 230 and 270.

With these coordinates, we have the following correspondences:

- (0,0) is a point which is right in front (at 1 m)
- (0,90) is a point which is right above (at 1 m)
- (0,180) is a point which is right behind (at 1 m)
- (90,0) is a point which is straight on the right (at 1 m)
- (-90,0) is a point which is straight on the left (at 1 m)

2.1 Problem 1: Binaural filtering

In this problem, we will program a function called hrtf_binaural.m which will simulate a binaural system (see figure 2.1.2). This program will be tested on several sound trajectories to be defined.

2.1.1 Binaural filtering for a fixed position in space

- Choose a subject number and load its corresponding HRIR datasets which are stored in a file .mat
- Use the provided program *GetNearestUCDpulse(azimuth,élévation,h3D)*, to obtain the HRIRs corresponding to the predefined fixed source position in space.
- load one of the sound example provided in the sound directory.
- Compute the binaural signals y_g and y_d by filtering the original signal with the corresponding HRIR for left and right ears and build the stereo signal. Listen and comment the results.

2.1.2 Binaural filtering for a trajectory

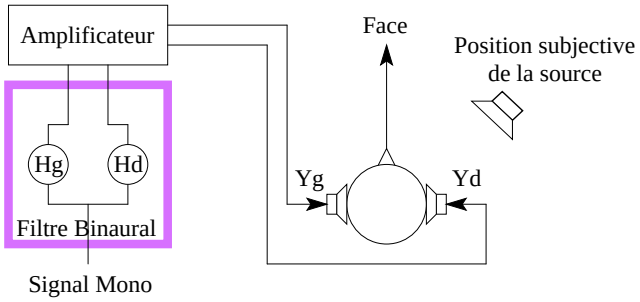


FIG. 2 – *binaural signal synthesis*

- Define a sound trajectory in space (e.g. on the sphere of radius 1 meter around the head)
- Synthesize and listen the sequence of sounds placed along this trajectory. For the sake of simplicity, choose a brief sound (for example 'snare1_44.wav') and play the sound at a single position until its end, and then re-play the sound at its new location, and so on...
- What could you do to have a sound that moves continuously in space?
- Compare this approach with the simple "panpot" [1]? What are the main differences? do you have a preference? are the provided HRTF optimal? Justify your answer and propose solutions to improve this system in general, and in particular for you.

2.2 Problem 2: Transaural filtering

Transaural filtering consists is transposing the previous system for a loudspeaker setting. To that aim, it is necessary to transform the binaural signal to take into account the contribution of each loudspeaker on each ear.

The figure 3 gives a simple implementation of binaural signal transcoding into a transaural signal.

The goal of this problem is then to program such a system.

- First, express the filters H'_{gg},H'_{dd},H'_{dg},H'_{gd} from the filters H_{gg},H_{dd},H_{dg},H_{gd} corresponding to the transfer functions between a given loudspeaker and one ear (fro instance H_{gd} corresponds to the transfer function between the left loudspeaker and the right ear)
- To realize the transaural system, one may inverse the filter $H_{den} = H_{gg}H_{dd} - H_{dg}H_{gd}$ using the least square technique. Let $G(z) = H(z)^{-1}$, we have $d(n) = h_{den}(n) * g(n) = \sum_{k=0}^{N-1} h_{den}(k)g(n-k)$. Then, writing this equation for n in the range [-A, C+2], one

1. we recall that the stereo panpot method consists in computing a stereo signal using the following formula and where $x(n)$ is the original monophonic signal:

$$D = (1-\alpha)x(n) \quad (1)$$
$$G = \alpha x(n) \quad (2)$$

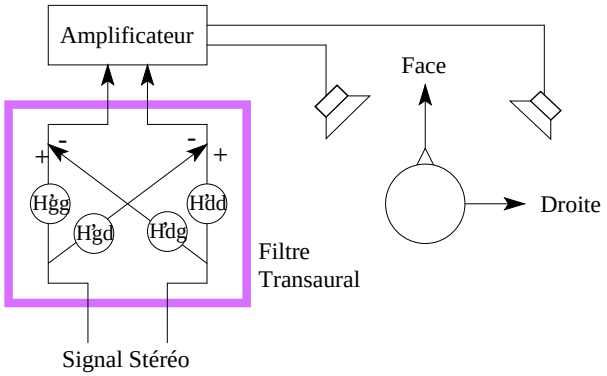


FIG. 3 – *Transaural rendering for a loudspeaker setting*

obtains (for instance for the case $N = 3$):

$$\begin{cases} d(-A) &= h(0)g(-A) \\ d(-A+1) &= h(1)g(-A) + h(0)g(-A+1) \\ d(-A+2) &= h(2)g(-A) + h(1)g(-A+1) + h(0)g(-A+2) \\ \vdots &= \vdots \\ d(0) &= h(2)g(-2) + h(1)g(-1) + h(0)g(0) \\ \vdots &= \vdots \\ d(C) &= h(2)g(C-2) + h(1)g(C-1) + h(0)g(C) \\ d(C+1) &= h(2)g(C-1) + h(1)g(C) \\ d(C+2) &= h(2)g(C) \end{cases} \tag{3}$$

To guarantee that $g(n)$ represents the impulse response of the inverse filter, it is necessary that $d(n)$ verifies $d(n) = (0,0,\ldots,0,1,0,\ldots,0)$. The previous equations can be rewritten in matricial form as $\mathbf{d} = \mathbf{Hg}$ where $\mathbf{H}$ is a Toeplitz matrix built from $h(0)$,$h(1)$,$\ldots h(N-1)$. A solution is then given by the pseudo inverse $\mathbf{g} = \mathbf{H}^{\dagger}\mathbf{d}$. For the matlab programming, one can use the function `toeplitz` and operator $\backslash$.