Digital Audio Signal Processing Lab Sessions

Session 1: Binaural synthesis and 3D audio.

Alexander Bertrand, Jesper Brunnström and Julian Schott.[1]

October 2022

Practical information

The code you have created to solve the exercises in the lab should be submitted by 12.00 (midday) the day before the evaluation session. Make sure any dependencies (functions or data) are included in the submission, such that it will run on any computer. For the same purpose, avoid absolute file paths in the scripts.

Please include a script called `main.m` which solves the selected exercise, Task 6 of Section 1-4: 3D Audio. The script should display information to understand how well the method performs the task. This should be in the form you deem most appropriate, for example terminal output, figures, audio, or a combination.

Preliminaries

Exercise 1-1: Simulate room impulse responses

This exercise is also included in H09M0A (P&D Information Systems and Signal Processing) and is repeated here for convenience. You will use a simulation environment to generate room impulse responses (RIRs) from a user-defined acoustic scenario.

1. Download the zip-file `sim_environment.zip` from the course website.
2. Extract the zip-file in a new folder in your home directory.
3. Open MATLAB and set your MATLAB path to this new folder.
4. Compile the included mex-file by typing the command `mex simroommex.c` in MATLAB.
5. Open the graphical user interface by typing `mySA_GUI`. The window that pops up allows you to define a set of parameters used by the simulation environment:
   - *Room dimension*: the size of the room in meters (the room has a square shape).
   - *T60*: the T60 reverberation time (in seconds) of the room. Set the T60 to zero to create a non-reverberant room.
   - *Length RIR*: the length of the room impulse responses that are simulated (in number of samples). Using shorter lengths results in a faster simulation time, but then you should make sure that significant parts of the reverberant tail of the RIR are not cut off.

[1]For questions and remarks: `jesper.brunnstroem@kuleuven.be` and `jschott@esat.kuleuven.be`.

- *Sampling frequency*: the frequency at which the RIRs are sampled. Large sampling rates yield more realistic results, but require more time to simulate.

Set the room size to 10 m, the T60 to 1.5 s, and keep the other parameters to their default values,

6. The simulation environment allows you to place a microphone array of n microphones in a linear configuration (with a vertical orientation). Choose the number of microphones equal to 3 in the field denoted by '# mics'. Choose an inter-microphone distance of d =5 cm in the field denoted by 'd [cm]'. Then click on the Mic-button, and click somewhere in the grid to define the position of the lower microphone in the array.

7. Click on the Audio-button to also place a target speech source in the room (at the left-hand side of the microphone array). Repeat this to place a second target source.

8. Click on the Noise-button to place a noise source in the room.

9. Click on 'Create/Store RIRs', and wait until a confirmation appears.

10. If you have done this correctly, a new mat-file `Computed_RIRs.mat` should appear in the current MATLAB directory. Open this mat-file and check which variables it contains. Try to find out what each variable represents (based on the user-defined values in the simulation environment). The RIRs are stored in the variables `RIR_source` and `RIR_noise`. Try to figure out what each dimension of these variables represents. How many RIRs have been generated, and why?

11. Create a new figure and plot the RIR from source 2 to microphone 3. Can you recognize the direct path component, the early reflections, and the reverberant tail?

Exercise 1-2: Generate microphone signals

This exercise is also included in H09M0A (P&D Information Systems and Signal Processing) and is repeated here for convenience. You will use the simulated RIRs to emulate the signals that are recorded by the microphone array.

1. Create an m-file with the name '`create_micsigs.m`'. This file will also be used for generating microphone signals in all other exercise sessions, hence it is essential that you pay particular attention to the parametrization of the code, i.e. the code should be able to function for changes in the number of microphones, target speech and noise sources, different sampling frequencies, etc. There are a few initialization steps to be performed within this m-file as follows:

   (a) Create a cell array to store the file names of the target speech sources (speech signals), e.g. speechfilename{1} = 'speech1.wav', speeechfilename{2} = 'speech2.wav', etc.

   (b) Repeat the same procedure as in (a) to store the file names of the noise sources (noise signals).

   (c) Set the desired length of the recorded microphone signals in seconds.

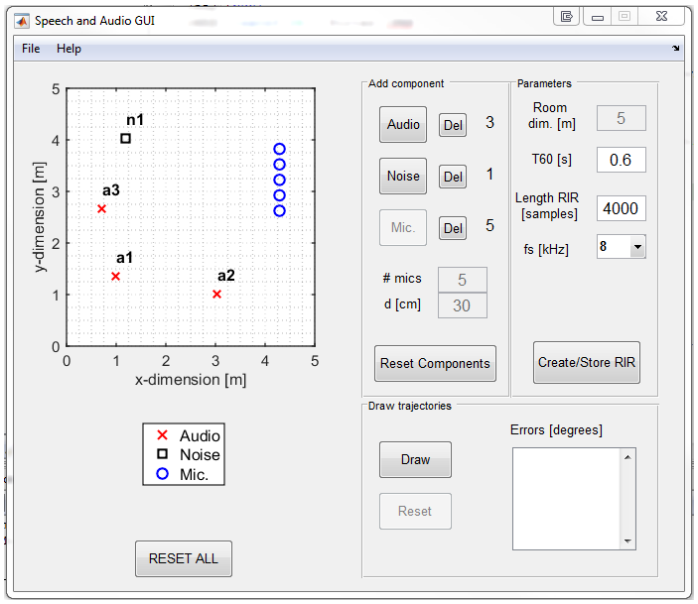

Figure 1: The room acoustics GUI.

(d) For each target speech source and noise source, resample them accordingly to match the sampling frequency of the RIR (not the other way around!).

The file should then perform the following tasks[I]:

- Read out the file `Computed_RIRs.mat`.
- Generate the n microphone signals that are recorded by the microphone array as defined in the simulation environment, i.e., containing the target speech + noise mixtures based on the RIRs stored in `Computed_RIRs.mat`.
- Put the n microphone signals in the columns of a matrix variable `mic`. Save this variable in a mat-file with the same name, together with the value of the RIR sampling rate.
- Plot the first two microphone signals in a single plot in different colors (using `hold on`).

Remark: When filtering a source signal with a RIR, make sure that the sampling frequencies match. Resample the source signal to the sampling rate of the RIR that was chosen in the GUI (not the other way around). Do this without manually hard-coding the sampling frequencies within the code (read out the sampling frequency of the wav files and use the variable `fs_RIR`).

2. Test the file '`create_micsigs.m`' in a non-reverberant scenario ($T60 = 0$ s) with a single target speech and noise source. Use the file `speech1.wav` for the target speech source, and use the file `Babble_noise1.wav` for the noise source (these files can be downloaded from the course website). Listen to the signal recorded by the first microphone using the command `soundsc`.

Remark: Note that the GUI contains several reset buttons to reset or redefine the values of certain parameters or the source/microphone locations. Also note that there is a save and load option to save a scenario for later use.

3. Repeat this procedure for the same sources-microphones configuration, but where the room dimension is set to 10 m and the reverberation time is set to 2 s (make sure your impulse response is sufficiently long). Listen to the first microphone signal and compare with the previous scenario. What do you conclude in terms of the speech intelligibility in highly reverberant scenarios?

4. Create a new scenario containing a 2-microphone array with $d = 1$ cm, and a single target speech source impinging on the microphone array at an angle of 45° (note that the microphone array has a vertical configuration). There is no noise source. Set the sampling frequency of the RIRs to 44.1 kHz, and set the reverberation time to 0 s (no reverberation). Try to predict what the RIRs for both microphones will look like, and in what sense the two microphone signals will differ from each other. Confirm by simulation.

5. Simulate the same scenario, but now with a sampling frequency of 8000 Hz. Make a new plot of the two microphone signals in a single figure (using `create_micsigs.m`). What do you see? Explain.

6. With the previous scenario, you have discovered a problem that may hamper the assessment of multi-microphone audio processing algorithms when using simulated audio signals based on simulated RIRs. Besides increasing the sampling frequency of the RIRs, what else can you do to reduce this effect?

Binaural synthesis and 3D audio

Consider the speech communication system depicted in Fig. 2. In a (noisy) recording room (left-hand side), the direction-of-arrival (DOA) of a target speech source is estimated (e.g. by means of the MUSIC algorithm), after which the speech signal is recorded and denoised by means of a beamformer. The speech signal is then played back in a listening room (right-hand side), where the original DOA of the target speech source is reproduced by means of a 3D audio effect using an array of loudspeakers. The system in the recording room is assumed to be in place and will not be implemented. Here, the focus will be on the listening room, i.e. generating the 3D audio effect by means of a loudspeaker array.

Exercise 1-3: Binaural synthesis

1. Write a new m-file `binaural_synthesis.m` to generate a binaural signal for headphones playback as follows:
   - Read out the wav file `speech1.wav`, and resample it to 8 kHz. Truncate the signal to a length of 10 s and store it in the variable `x`.
   - Make two copies of `x` and store both of them in the columns of a 2-column matrix variable `binaural_sig1_1=[x x]`, which will serve as a binaural signal where the first column is played back at the left ear, and the second column is played back at the right ear.
   - Create a second binaural signal as `binaural_sig2_1=[x 0.5*x]`, i.e., decrease the amplitude of the right channel (second column) with 50%.
   - Create a third binaural signal `binaural_sig3_1`, which is a copy of `binaural_sig1_1`, but add a delay of 3 samples in the right channel.
   - Create a fourth binaural signal `binaural_sig4_1` by filtering the left and the right channel of `binaural_sig1_1` with the filters in the first and second column of the matrix variable in `HRTF.mat`, respectively. The two filters in `HRTF.mat` are measured head related transfer functions (HRTFs [2]) for the left and the right ear.
   - Repeat this procedure for the second wav file (`speech2.wav`), to generate four new binaural signals `binaural_sig1_2`, `binaural_sig2_2`, `binaural_sig3_2`, and `binaural_sig4_2`, but this time reverse the left and right channels, e.g., `binaural_sig2_2` is then defined as `binaural_sig2_2=[0.5*x x]`.
   - Generate `binaural_sig1=binaural_sig1_1+binaural_sig1_2`, and proceed similarly for `binaural_sig2`, `binaural_sig3`, and `binaural_sig4`.
2. Using headphones or earphones, listen to the four created binaural signals `binaural_sig1`, `binaural_sig2`, `binaural_sig3`, and `binaural_sig4`. Make sure that the left channel is played back by the left loudspeaker and the right channel by the right loudspeaker (use the command `soundsc`). Which effect do you observe? Pay special attention to `binaural_sig3` (note that, remarkably enough, there was no manipulation in the amplitude of the signal (!)).
3. Explain the effect of the different manipulations that have been used to create `binaural_sig2`, `binaural_sig3`, and `binaural_sig4`.

[2] `https://en.wikipedia.org/wiki/Head-related_transfer_function`

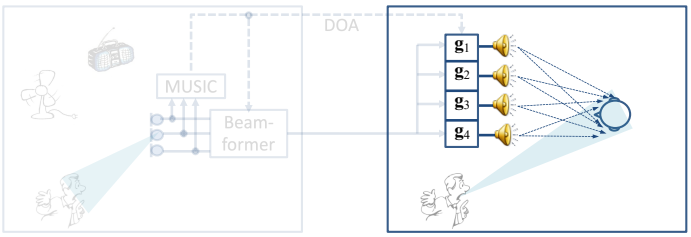

Figure 2: Speech communication with focus on 3D audio effects.

4. After listening to `binaural_sig4`: can you make a rough estimate of the DOA of the sound source that was used to measure the HRTFs? Was it measured in a reverberant room or in an anechoic room?

Exercise 1-4: 3D audio

The goal of this exercise is to design the FIR filters $\mathbf{g}_j[t]$ in Fig. 2 such that the listener has the impression that the signal played by the loudspeaker array is coming from a given target direction. Let $X_L(z)$ and $X_R(z)$ denote the left and right HRTF corresponding to this target direction, then the filters $G_j(z)$, $j = 1 \dots J$ (in the z-domain) are designed such that (explain!)

$$\begin{bmatrix} H_{1L}(z) & H_{2L}(z) & \dots & H_{JL}(z) \\ H_{1R}(z) & H_{2R}(z) & \dots & H_{JR}(z) \end{bmatrix} \begin{bmatrix} G_1(z) \\ G_2(z) \\ \vdots \\ G_J(z) \end{bmatrix} = \begin{bmatrix} X_L(z) \\ X_R(z) \end{bmatrix} \tag{1}$$

where $H_{jL}(z)$ and $H_{jR}(z)$ denote the acoustic transfer function from loudspeaker j to the left and right ear of the listener, respectively.

In the sequel, it will be assumed that the RIRs corresponding to $H_{jL}(z)$ and $H_{jR}(z)$, $j = 1, \dots, J$, are known, e.g., by using training sequences during a calibration phase (this is beyond the scope of this exercise).

1. Using pen and paper, formulate the discrete time-domain representation of (1), where it is assumed that all filters are FIR. Use a matrix description of the different convolution operations by replacing $G_j(z)$ with a vector of length L_g, and $H_{jL}(z)$ with a $(L_h + L_g - 1) \times L_g$ Toeplitz matrix, where L_h denotes the length of the RIR corresponding to $H_{jL}(z)$, and where L_g denotes the length of the (unknown) FIR filter corresponding to $G_j(z)$ (it is assumed that L_g and L_h are independent of j and L/R) [II]. Keep the paper with the derivation for the evaluation session.

2. The time-domain representation of (1) results in a system of equations (SOE)

$$\mathbf{H}\mathbf{g} = \mathbf{x} \tag{2}$$

in the unknowns $\mathbf{g} = [\mathbf{g}_1^T \ \ldots \ \mathbf{g}_J^T]^T$, where $\mathbf{g}_j$ is a vector of length L_g. Derive a condition for the (minimum) value for L_g as a function of J and L_h, such that the SOE (2) can be solved **exactly**. What is the minimum number of loudspeakers J required to create a perfect 3D audio impression? [III]

3. The HRTFs $X_L(z)$ and $X_R(z)$ are modeled as FIR filters. However, in the SOE (2), a delayed version of the HRTFs is typically included, i.e., $z^{-\Delta}X_L(z)$ and $z^{-\Delta}X_R(z)$, with Δ a pre-defined delay. Explain why this delay is added. How do you have to modify the right-hand sid (RHS) of (2) to include such a delay? [IV]

4. The signals impinging at the left and right ear of the listener in Fig. 2 are generated using RIRs, computed by `mySA_GUI.m`. To this end, create a microphone array with two microphones with an inter-microphone distance of 15cm (these microphones represent the two ears of the listener). Add $J = 5$ sound sources around the microphone array, and make sure that all of them are at a similar distance from the microphone array (these will represent the five loudspeakers). Set $T_{60} = 0.5$ s and set the sampling rate to 8 kHz. Store the resulting impulse responses, and use them in the sequel.

5. Complete the missing sections in the Matlab code provided, `SOE_skeleton.m`, which implements the calculation of the filters $\mathbf{g}_j$ from the SOE. The script should perform the following tasks:

   - Load `Computed_RIRs.mat` and set the length of your RIRs to 400 for now (this is to obtain a reasonable computation time when solving (2)).
   - Create two vector variables `xL` and `xR` that represent two HRTFs, either simulated or measured respectively. Firstly define both HRTFs as the one-tap filter `[1]`, to simulate a target source in the frontal direction. In the sequel it is assumed that the length of the HRTFs is much smaller than L_h (note that this may require a truncation of the HRTFs if this assumption is not satisfied).
   - Construct the matrix $\mathbf{H}$ and the vector $\mathbf{x}$ of the SOE (2). Do not forget to add the delay Δ (see part 3 of this exercise). A relevant value for this delay is:

     ```
     Delta=ceil(sqrt(room_dim(1)^2+room_dim(2)^2)*fs_RIR/340) .
     ```

   - Remove the all-zero rows in `H`, and the corresponding elements in `x`, and then solve the SOE (2) [V]. Does the removal of the all-zero rows in `H` influence the solution? Why (not)?

   - Create a figure that plots the result of `H*g` (in blue), together with the vector `x` (in red), and print the value `synth_error=norm(H*g-x)` in the MATLAB workspace.
   - Read out the wav file `speech1.wav`, and resample it to 8 kHz. Truncate the signal to a length of 10 s.
   - Create a 2-column matrix variable `binaural_sig` which contains the two signals observed by the left and right ear of the listener in Fig. 2, when the array of 5 loudspeakers defined with `myGUI_SA.m` plays the speech signal `speech1.wav` filtered with the 5 synthesis filters stacked in `g` (do not forget to resample the speech signal to 8 kHz).

6. Test your code on the scenario mentioned earlier in task 4 (and listen to the binaural signal `binaural_sig`, it should not be distorted). You should obtain a `synth_error` smaller than 10^{-10}.

7. Re-run the experiment, but set `length RIR` in the GUI to 1500 taps (using larger values is at your own risk...). Note that the computation time may become quite large (up to 1-2 minutes). You should obtain a `synth_error` smaller than 10^{-10}.

8. Manipulate the HRTFs `xL` and `xR` in order to recreate the binaural signals from Exercise 1-3, and listen to the results. Do you hear similar effects?

9. Can you create a target source coming from the left-hand side of the listener if all the loudspeakers are on the RHS side of the listener (as in the setup of Fig. 2)? Why (not)?

10. Is the sweetspot, i.e., the region in which the listener can move while still hearing the correct 3D audio effect, larger or smaller than 5 cm? How can you test this? The code should perform this test based on the value of the boolean flag `sweetspotFlag`.

11. Add white noise to the elements in `H` with a standard deviation of 5% of the standard deviation of the elements in the first column of `H`. The code should perform the addition based on the value of the boolean flag `noiseFlag`. This noise models estimation errors during the calibration phase when identifying the RIRs. Re-run the experiment with these noisy RIRs, and analyze the outcome (i.e. design `g` based on the noisy RIRs and analyse the binaural signals with the exact RIRs). What does this tell you about the sensitivity of the procedure with respect to modeling errors?

Hints

[I] Check the help file of the following commands, which may be useful in the above task: `audioread, fftfilt, resample, load, save`.

[II] Remember that the discrete linear convolution between two discrete signals $a(i)$ and $b(i)$, respectively with support (i. e. the smallest interval such that the signal is zero outside the interval) $[1, L_a]$ and $[1, L_b]$, with $L_b > L_a$, can be defined using a matrix multiplication between the Toeplitz matrix containing the elements of $a(i)$ and the vector containing the elements of $b(i)$:

$$a(i) * b(i) = \begin{bmatrix} a(1) & 0 & \dots & 0 \\ a(2) & a(1) & \ddots & \vdots \\ \vdots & a(2) & \ddots & 0 \\ a(L_a) & \vdots & \ddots & 0 \\ 0 & a(L_a) & \vdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a(L_a) \end{bmatrix} \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(L_b) \end{bmatrix} \tag{3}$$

[III] Think about the behavior of a system of equations based on the relationship between the number of equations and the number of unknowns.

[IV] Pay special attention to the first equation and the $(L_h + L_g)$-th equation of the SOE and assume, e.g., that both HRTFs are equal to $\mathbf{e}_1 = [1\ 0\ \dots\ 0]^T$, i.e., the target source is in the frontal direction. If all the loudspeakers are closer to the left ear than to the right ear, a problem will occur in the equations. Another problem will occur if there is a loudspeaker that is closer to both ears of the listener compared to all other loudspeakers. Try to sketch what the SOE looks like in one (or both) of the described situations to simplify the analysis of this problem.

[V] Use `g=H\x` in MATLAB. Do not worry if you obtain a warning for rank deficiency.