P&D Information Systems and Signal Processing

Week 1 - February 10-14, 2020
Acoustic simulation environment and direction-of-arrival estimation

Giuliano Bernardi, Randall Ali, Santiago Ruiz[1], Marc Moonen

Version 1.0

Prelimenary remarks:

- If you are not familiar with MATLAB, you should first go through the MATLAB tutorial, which is available in Toledo. It is recommended to first make all exercises in this tutorial. Ask your TA for help if you run into problems. If you are already familiar with MATLAB, you can skip the tutorial.
- Work in m-files, and save your work regularly, in order to be able to repeat your experiments later on. Do **not** save on the local hard-drive, since this will be erased each time you log off or when the computer crashes.
- Parameterize your code i.e. identify the main values that define a scenario or algorithm and name them as variables (parameters).
- All the supplementary files (see below) are available in Toledo.

Part 1: Room impulse responses

In this first part, an acoustic simulation environment is set up to generate room impulse responses (RIRs) from a user-defined 2-D acoustic scenario.

1. Download the zip-file `sim_environment.zip` from Toledo.
2. Extract the zip-file in a new folder in your home directory. Open MATLAB, and set the MATLAB path to this new folder.
3. Compile the included mex-file by typing `mex simroommex.c` in MATLAB.
4. Open the graphical user interface (GUI) by typing `mySA_GUI`. The window that pops up allows you to define a set of parameters used by the simulation environment, namely the room dimension, i.e. the size of the room in meters (the room has a square shape), the T60 reverberation time of the room in seconds (set the T60 to zero to create a non-reverberant room), the length of the generated RIRs in number of samples (using shorter lengths results in a faster simulation time, but then significant parts of the reverberant tail of the RIRs may be cut off), and the sampling frequency i.e. the frequency in Hz at which the RIRs are sampled (large sampling rates yield more realistic results, but require more time to simulate).

[1]For questions and remarks: `santiago.ruiz@esat.kuleuven.be`

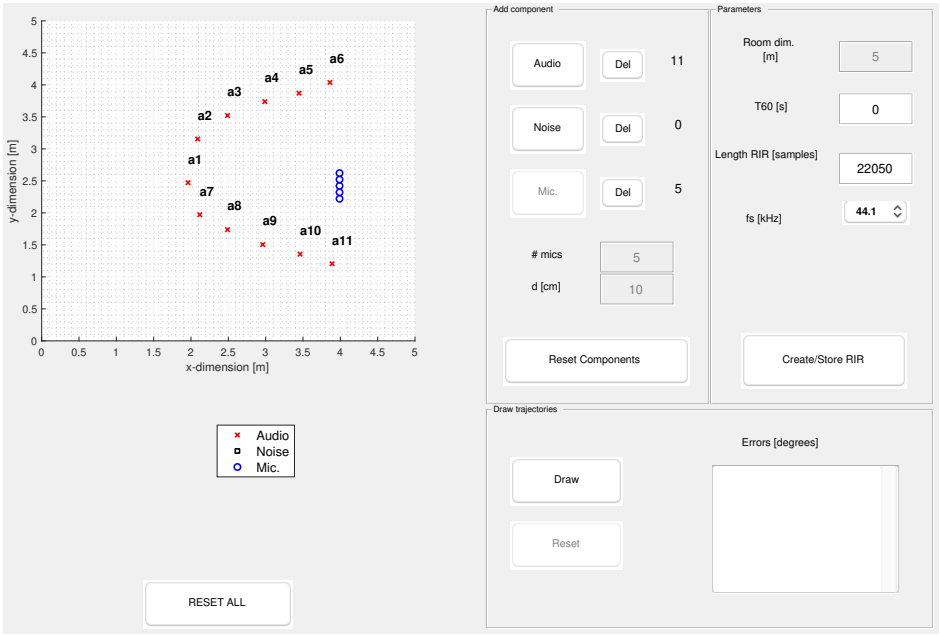

Figure 1: The room acoustics GUI.

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

5. The simulation environment allows you to place a microphone array of m 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 equal to 5cm 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.

6. Click on the Audio-button to place a target audio source in the room. The target source(s) should always be on the left-hand side of the microphone array. Repeat this to place a second target audio source.

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

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

9. 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 and what each variable represents (based on the user-defined values in the simulation environment).

10. Plot the RIR from source 2 to microphone 3. Identify the direct path component, the early reflections, and the reverberant tail. When plotting a result when the GUI is active, make sure to first open a new figure using the command `figure`. If not, the plot will be drawn in

the GUI window.

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.

Part 2: Microphone signals

In this part, the simulated RIRs are used to generate the microphone signals that are recorded by the microphone array.

1. Create an m-file with the name '**create_micsigs.m**'. 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 audio and noise sources, different sampling frequencies, etc. The initialization steps to be performed within this m-file are as follows:

   - Create a cell array that sets the file names that will be used for the target audio sources (speech signals), e.g. speechfilename{1} = 'speech1.wav', speeechfilename{2} = 'speech2.wav', etc.
   - Repeat the same procedure as in (a) to set the file names of the noise sources.
   - Set the desired length of the recorded microphone signals in seconds.
   - For each target audio source and noise source, re-sample them accordingly to match the sampling frequency of the RIRs (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`).

   The file should then perform the following tasks:

   - Read out the file `Computed_RIRs.mat`.
   - Generate the m microphone signals that are recorded by the microphone array as defined in the simulation environment, i.e., containing the target audio+noise mixtures based on the RIRs stored in `Computed_RIRs.mat`.
   - Put the m 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 sampling frequency.
   - Plot the first two microphone signals in a single plot (using `hold on`).

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

2. Test the file '`create_micsigs.m`' in a non-reverberant scenario (T60 = 0s) with a single target audio and noise source. Use the file `speech1.wav` for the target audio source, and use the file `Babble_noise1.wav` for the noise source (these files are available in Toledo). Listen to the first microphone signal using the command `soundsc`.

3. Repeat this procedure for the same source-microphone configuration, but where the room dimension is set to 10m and the reverberation time is set to 2s (make sure your impulse response is sufficiently long). Listen to the first microphone signal and compare with the previous scenario.

4. Create a new scenario with a 2-microphone array with $d = 1$cm, and a single target audio source impinging in a 45° angle on the microphone array. There is no noise source. Set the sampling frequency of the RIRs to 44.1kHz, and set the reverberation time to 0s. 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 8000Hz. Make a new plot of the two microphone signals in a single figure (using `create_micsigs.m`). With this, a problem is discovered 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 be done to reduce this effect?

Part 3: Cross-correlation-based TDOA estimation

1. Create a new scenario with a 2-microphone array with $d = 10$cm, and a single target audio source (no noise source) in a 10m room with a reverberation time of 0s. Use a sampling frequency of 16kHz to generate the RIRs. Put the source in the bottom-left region of the microphone array, close to a 135° angle with respect to the vertical axis of the microphone array (consider directly above the microphone array as 0°). Make sure there is a few meters distance between the source and the microphone array.

2. Create a new m-file **TDOA_corr.m** that performs the following tasks:

   - Identify the sample delay between the direct path components of the two RIRs. This will serve as ground truth for the estimation of the time-difference-of-arrival (TDOA).
   - Generate the two microphone signals from the above scenario where the target audio source is a white noise signal (re-use code from `create_micsigs.m`).
   - Estimate the TDOA from the two microphone signals, based on their (time-domain) cross-correlation function. Do this by shifting a representative segment of one microphone signal over the other microphone signal and identify the maximum cross-correlation peak. What trade-off can be made here in terms of the length of this segment?
   - Make a plot of the time-domain cross-correlation function. Indicate where the maximum peak is expected according to the ground truth (e.g., using `stem`).
   - Print the value of the difference between the estimated TDOA and the ground truth TDOA in the MATLAB workspace.

3. Test the file in the above non-reverberant scenario.

   Only if you have a zero TDOA estimation error, you can continue with the next step.

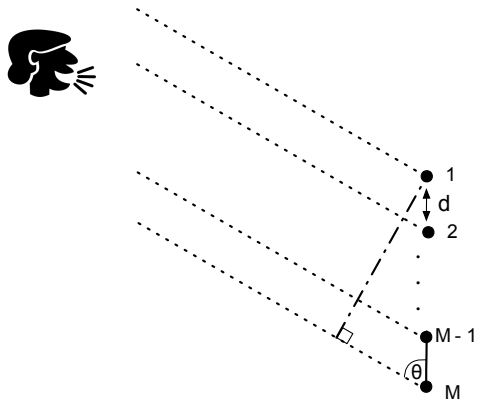

Figure 2: Schematic depiction of a scenario where the target audio source is in the far-field. In such a case, the DOA is θ. Based on this far-field model depiction the TDOA estimate can be converted into a DOA estimate, using d (the inter-microphone distance), m (the number of microphones) and c (the speed of sound in air).

4. Increase the reverberation time and check how much reverberation is allowed before the TDOA estimation starts to show errors.

5. Re-simulate the non-reverberant scenario, and replace the white noise signal by a speech signal in `TDOA_corr.m` (use the speech signal in `part1_track1_dry.wav`). Check how the shape of the cross-correlation function changes compared to the white noise case. Note that the speech signal is not continuously active i.e. be careful when selecting a representative signal segment to compute the cross-correlation function.

   Only if you have a zero TDOA estimation error, you can continue with the next step.

6. Again increase the reverberation time. Check if you can allow as much reverberation as in the white noise case.

Part 4: Cross-correlation-based DOA estimation

1. Create a new m-file **DOA_corr.m** that first estimates the TDOA as in Part 3, but then also transforms this estimate to a direction-of-arrival (DOA) estimate using the far-field assumption (see Fig. 1). Use a value of $c = 340$m/s for the speed of sound in air. The file should store the DOA estimate in a variable `DOA_est` and store this variable in a mat-file with the same name. The DOA estimate should be between 0-180°, where 90° corresponds to the so-called broadside direction to the *left* of the microphone array, and where 180° corresponds to the so-called end-fire direction where the source is *below* the microphone array.

2. Reconstruct the first (non-reverberant) scenario of Part 2, with a sampling frequency of

44.1kHz and a DOA of approximately 135°. Run the `DOA_corr.m` file, using a speech signal for the target audio source.

3. When pressing the 'Draw' button in the GUI, the GUI reads out the `DOA_est.mat` file and plots the DOA on the screen to visualize how much the estimated DOA is off target.[2] It will also show the DOA estimation error in a separate window. Check how well the estimated DOA matches with the true DOA.

4. Repeat the DOA computation for a scenario with more reverberation and check if the DOA estimate still accurate. If not, what can be done to reduce the error?

Part 5: Multi-source TDOA/DOA estimation

Create a new non-reverberant scenario at 44.1kHz with a 5-microphone array with d =10cm, two target audio sources, one should be in the bottom-left region (i.e. anywhere between 90° and 180°) from the microphone array, and the other in the top-left region (i.e. anywhere between 0° and 90°).

1. For each of the target audio sources, use a different white noise signal (`whitenoise_sig_1.wav` and `whitenoise_sig_2.wav`).

2. Re-use the code in `TDOA_corr.m` to compute and plot the cross-correlation function between any two of the microphone signals in a scenario where both sources are *simultaneously active*. Based on the observation of the corresponding cross-correlation functions, estimate the DOAs of both sources.

3. Increase the reverberation of the acoustic environment and estimate the DOAs again. Check how much reverberation can be tolerated until the DOA estimation becomes unreliable.

4. Repeat 1-3 with different target audio source signals. For the bottom-left source, use the signal `part1_track1_dry.wav`, and for the top-left source, use the signal `part1_track2_dry.wav`.

Part 6: Head-mounted microphones

Fig. 3 illustrates a setup that was used to measure 6×4 impulse responses from the azimuthal direction of each of the loudspeakers to each of the head-mounted microphones on a mannequin head. There are 4 microphones in total, 2 pertaining to the right ear, $\mathbf{y_{R1}}, \mathbf{y_{R2}}$ with a microphone spacing of 1.3cm and another 2 pertaining to the left ear also with a microphone spacing of 1.3cm, $\mathbf{y_{L1}}, \mathbf{y_{L2}}$. The spacing between the left and right ear is approximately 21.5cm.

These impulse responses are available in the folder `Head_mounted_IRs`, which contains a subfolder of impulse responses corresponding to the particular angle as depicted in Fig. 3. The impulse responses for $\mathbf{y_{L1}}, \mathbf{y_{L2}}$, $\mathbf{y_{R1}}$, and $\mathbf{y_{R2}}$ correspond respectively to the following wav files: `HMIR_L1.wav`, `HMIR_L2.wav`, `HMIR_R1.wav`, and `HMIR_R2.wav`.

[2] In a scenario with more than one target audio source, the number of estimated DOAs in the file `DOA_est.mat` must match the number of target audio sources placed in the room.

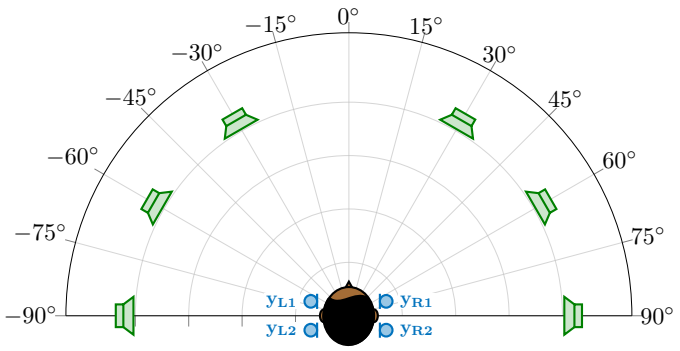

Figure 3: Setup for head-mounted microphones.

1. For each of the angles shown in Fig. 3, perform the following tasks:
   - Create a new set of microphone signals with **create_micsigs.m** and `part1_track1_dry.wav` as the target audio signal. Use the soundsc command, `soundsc([yL1 yR1], fs)`, to listen to a binaural signal.
   - Using the cross-correlation-based method to estimate the DOA using the following sets of microphone signals:
     - The two microphones pertaining to the right ear, $\mathbf{y_{R1}}, \mathbf{y_{R2}}$
     - The two microphones pertaining to the left ear, $\mathbf{y_{L1}}, \mathbf{y_{L2}}$
     - The two frontal microphones, $\mathbf{y_{L1}}, \mathbf{y_{R1}}$.
2. Check if the performance depends on the set of microphones used. What changes can be made make in order to improve the performance?
3. Repeat the experiments in (4) from Part 5 using these measured impulse responses from the head-mounted microphones as opposed to the simulated RIRs. Generate a set of microphone signals with `part1_track1_dry.wav` for the target audio source coming from any direction on the left-hand side and `part1_track2_dry.wav` coming from any direction on the right-hand side. Listen to the binaural signal. Estimate the DOA of each of the sources as accurately as possible.